

POLITECNICO DI MILANO
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione



**M.O.E. : UN SISTEMA DI VISIONE
STEREOSCOPICO BIOISPIRATO PER
IL ROBOT MAXIMUM ONE**

AI & R Lab
**Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano**

Relatore: Prof. Giuseppina Gini
Correlatore: Ing. Flavio Mutti

Tesi di Laurea di:
Giacomo Giacchetti, matricola 680825
Renzo Vannucci, matricola 708568

Anno Accademico 2009-2010

Ai nostri cari...

Sommario

Il presente elaborato di tesi tratta la progettazione e la realizzazione prototipale di una testa robotica per il robot Maximum One, che chiameremo Maximum One hHead o brevemente MOE. L'obiettivo prefisso è quello di emulare tramite MOE quelle che sono le caratteristiche del sistema visivo umano. Nell'elaborato trattiamo lo stato dell'arte delle teste robotiche, lo sviluppo della parte meccanica, della parte elettronica ed infine delle librerie software per il controllo del sistema realizzato. Concludiamo la trattazione descrivendo i test effettuati e commentando i risultati ottenuti evincendone pregi (quali affidabilità e precisione) e difetti (quali basse velocità di rotazione in certe condizioni e peso eccessivo). E' presente anche una sezione sui possibili sviluppi futuri che, a nostro avviso, potrebbero migliorare le caratteristiche del prototipo.

Ringraziamenti

Ringraziamo in modo particolare la Professoressa Gini per l'interessante lavoro di tesi che ci ha permesso di sviluppare. Il nostro correlatore, per la pazienza che ha dimostrato nel saper ascoltare le nostre domande, nell'aiutarci a trovare gli errori e le possibili soluzioni, e soprattutto la passione per il proprio lavoro, doti molto rare e preziose, grazie Flavio. Il Signor Vannucci per i consigli sulla parte hardware della tesi. I ragazzi dell'AirLab per i numerosi consigli che hanno saputo darci nel corso di questi mesi, in particolare Davide Rizzi e Martino Migliavacca.

Giacomo e Renzo

Vorrei ringraziare coloro che hanno saputo credere in me, senza di loro il percorso che mi ha portato fino a qui sarebbe stato sicuramente più difficile. Ringrazio i miei genitori, che mi hanno sostenuto in questi anni, esortandomi anche e soprattutto nei momenti di difficoltà. Un immenso grazie a Susanna, compagna durante questo mio percorso universitario, per la tua presenza ed affetto costante lungo questo impegnativo cammino. Il pensiero va anche a Salvo, mio coinquilino e caro amico, ed i suoi fratelli Davide e Giuseppe, con i quali ho trascorso gli ultimi mesi di lavoro. Un ringraziamento particolare va al signor Pagani per i preziosi consigli e per la disponibilità nella creazione dei pezzi di cui avevo bisogno. Grazie anche ad Alessandro per i disegni CAD dei prototipi ed a Mattia per i consigli sulle procedure burocratiche riguardanti la tesi, avrei dimenticato sicuramente qualcosa senza il tuo aiuto. Di nuovo grazie a tutti per i bei momenti passati insieme.

Giacomo

Ringrazio innanzitutto mio padre per l'aiuto tecnico e la motivazione che ha saputo fornirmi. In generale un grazie va alla mia famiglia per la pazienza dimostrata. Alla mia ragazza per essermi stata vicino.

Renzo

Indice

Sommario	I
Ringraziamenti	III
1 Introduzione	1
2 Stato dell'arte	5
2.1 Teste robotiche	5
2.1.1 Probo	5
2.1.2 Icube	6
2.1.3 i-RoK	7
2.2 Sistemi visivi	10
2.2.1 Telecamere attuate tramite PAM	10
2.2.2 Sistemi Pan-Tilt	11
3 Specifiche richieste	13
3.1 Specifiche bioispirate	13
3.1.1 Campo visivo	13
3.1.2 Proporzioni	14
3.1.3 Velocità di rotazione	14
3.2 Specifiche non bioispirate	14
3.2.1 Risoluzione e frame rate	14
3.2.2 Elevata precisione	14
3.2.3 Peso	14
4 Bioispirazione	17
4.1 Evoluzione della robotica e biorobotica	17
4.1.1 Bionica, biorobotica, biomimetica e robotica bioispirata	18
4.2 Caratteristiche visive umane	19
4.2.1 L'occhio umano	19
4.2.2 I muscoli dell'occhio	21

4.2.3	Visione e campo visivo umano	22
4.3	Classificazione dei movimenti oculari	22
4.3.1	Inseguimento lento	23
4.3.2	Inseguimento veloce - Saccadi	23
4.3.3	Vergenza	24
4.4	Caratteristiche visive emulate nel sistema MOE	25
4.4.1	Proporzioni	25
4.4.2	Campo visivo	25
4.4.3	Velocità di rotazione	26
5	Sistema meccanico MOE	29
5.1	Prototipi preliminari	29
5.1.1	Prototipo Nr.1	29
5.1.2	Prototipo Nr.2	31
5.1.3	Prototipo Nr.3	32
5.2	Prototipi costruiti	33
5.2.1	Prototipo Nr.4	33
5.2.2	Prototipo Nr.5	35
5.2.3	Prototipo Nr.6	36
5.2.4	Prototipo Nr.7	37
6	Architettura dell'apparato elettronico MOE	41
6.1	Introduzione	41
6.2	Componenti integrati utilizzati	42
6.2.1	PIC 16F877	42
6.2.2	MAX232	43
6.2.3	LM7805	43
6.2.4	Motori passo-passo	44
6.2.5	L297	47
6.2.6	L298 e pilotaggio in PWM	52
6.3	Realizzazione apparato elettronico MOE	54
6.3.1	Schema elettrico scheda microprocessore	54
6.3.2	Schema elettrico della scheda di pilotaggio	57
7	Firmware MOE	59
7.1	Introduzione	59
7.2	RS232	60
7.2.1	Modulo USART nel PIC16F877	62
7.3	Protocollo di comunicazione	66
7.4	Comandi implementati	69

7.5	Gestione dello stato	70
7.5.1	Interrupt nel PIC16F877	72
8	Libreria MOE	77
8.1	Introduzione	77
8.2	Libreria	77
8.3	libmoe e tabelle di velocità	78
8.3.1	Makefile e grafo dipendenze	80
8.4	MOEtest	81
8.4.1	Synclient	81
8.4.2	Utilizzare libmoe	84
9	Valutazioni sperimentali MOE	89
9.1	Test sulle posizioni	89
9.2	Test di velocità	90
9.3	Test di precisione	91
9.3.1	Raccolta dei dati	93
10	Conclusioni e sviluppi futuri	103
10.1	Conclusioni	103
10.1.1	Punti di forza del sistema MOE	104
10.1.2	Punti deboli del sistema MOE	104
10.2	Possibili miglioramenti	105
10.2.1	Sostituzione motore verticale	105
10.2.2	Controllo in microstepping	105
10.2.3	Miglioramento della parte meccanica	106
10.3	Sviluppi futuri e varianti	106
10.3.1	Muscoli di McKibben	106
10.3.2	Integrare il movimento del collo	107
10.3.3	Computer vision	107
10.3.4	Nuovi test	109
	Bibliografia	110
A	Il manuale utente	115
A.1	Connessione dei cavi	115
A.2	Documentazione libmoe	122
A.2.1	parse.csv.h	122
A.2.2	utils.h	123
A.2.3	command.h	124

Elenco delle figure

2.1	Occhi del robot probo	6
2.2	Immagine del robot probo	6
2.3	Testa robotica di i-cub	7
2.4	Meccanismo utilizzato per gli occhi di i-cub	7
2.5	Testa robotica i-RoK	7
2.6	Misure di i-RoK in mm	7
2.7	Struttura che permette il movimento oculare di i-RoK	9
2.8	Telecamera Attuata da muscoli artificiali	10
2.9	Sistema di PAM antagonisti	11
2.10	Sistema pan and tilt Headroid	12
3.1	Primo piano del collo del robot Maximum One	15
4.1	La struttura dell'occhio umano	19
4.2	L'occhio come una macchina fotografica	19
4.3	L'anatomia della retina	20
4.4	Rapporto gangli / fotorecettori	20
4.5	I muscoli dell'occhio	21
4.6	Campo visivo umano	22
4.7	Estensione orizzontale del campo visivo di MOE	25
4.8	Estensione verticale del campo visivo di MOE	25
5.1	Rappresentazione frontale semplificata del prototipo nr.1	30
5.2	Rappresentazione frontale semplificata del prototipo nr.2	31
5.3	Rappresentazione frontale semplificata del prototipo nr.3	32
5.4	Rappresentazione frontale semplificata del prototipo nr.4	33
5.5	Parte bassa della struttura mobile del prototipo nr.4.	34
5.6	Rappresentazione frontale semplificata del prototipo nr.5	35
5.7	Rappresentazione frontale semplificata del prototipo nr.6	36
5.8	Rappresentazione frontale semplificata del prototipo nr.7	37

5.9	Primo piano del sensore di finecorsa montato vicino alla telecamera destra.	38
5.10	Fotografia della versione definitiva della struttura meccanica di MOE.	39
6.1	Interfacciamento RS232 tramite MAX 232	44
6.2	Schema elettrico LM7805	44
6.3	Motore passo-passo bipolare	45
6.4	Pilotaggio one phase on	45
6.5	Pilotaggio two phase on	46
6.6	Pilotaggio half step	46
6.7	Circuito chopper per pilotaggio PWM	48
6.8	Diagramma pin L297	49
6.9	Diagramma stati pilotaggio di un passo-passo	51
6.10	Circuito ideale ponte H	52
6.11	Ponte H, nel L298	53
6.12	Pilotaggio tradizionale e pilotaggio PWM	53
6.13	Schema elettrico scheda microprocessore	55
6.14	Schema elettrico sensore ottico	57
6.15	Schema elettrico pilotaggio motore passo-passo	58
7.1	Diagramma PIN PIC16F877	63
7.2	Registro RCSTA	64
7.3	Registro TXSTA	65
7.4	Diagramma stati Parser	67
7.5	Diagramma gestione dello stato	70
7.6	Registro TOIF	75
8.1	Esempio di tabella di velocità	79
8.2	Grafo dipendenze oggetti libmoe	80
8.3	Output synclient	83
9.1	Matrice delle immagini	94
9.2	Estrazione degli angoli dalla scacchiera	94
9.3	Intorni degli angoli della scacchiera	94
9.4	Distribuzione degli errori	95
9.5	Parametri estrinseci rispetto alla scacchiera	95
9.6	Parametri estrinseci rispetto alla camera	95
A.1	Nessun cavo collegato	116
A.2	Cavi	117

A.3	Collegamento motori	118
A.4	Collegamento sensori	119
A.5	Collegamento completato	120
A.6	Collegamento tester	121

Elenco delle tabelle

6.1	Caratteristiche di base del PIC 16F877-20/P	42
6.2	Specifiche motore RS535-0489	47
6.3	Specifiche motori RS440-420	47
6.4	Tabella sulla descrizione dei pin dell'integrato	50
7.1	Tabella fornita dalla Microcip	62
7.2	Informazioni contenute nel msg_rcvd	67
7.3	Estratto della tabella di velocità	72
9.1	Velocità di rotazione raggiungibili in base all'angolo di spostamento	91
9.2	Test eseguiti	92
9.3	Test eseguiti sul motore numero 1 processando le immagini scattate con la telecamera sinistra	98
9.4	Test eseguiti sul motore numero 2 processando le immagini scattate con la telecamera sinistra	99
9.5	Test eseguiti sul motore numero 3 processando le immagini scattate con la telecamera destra	100
9.6	Indici di sintesi degli errori medi e massimi	101

Capitolo 1

Introduzione

*“I libri hanno gli stessi nemici degli uomini:
il fuoco,
l’umido,
le bestie,
il tempo
ed il loro stesso contenuto.”*

Paul Valéry (1871-1945), poeta francese ...

Nell’ambito della robotica sono di grande importanza lo studio e lo sviluppo di sistemi automatici volti all’analisi ed all’interazione con l’ambiente esterno. La modalità sensoriale che in questi sistemi viene elaborata per prima è senza dubbio la percezione visiva. In questo elaborato verrà descritta la realizzazione di una testa robotica per il robot Maximum One, capace di ricevere informazioni tramite una coppia di telecamere mobili. Il sistema denominato M.O.E. come acronimo di Maximum One hEad coinvolgerà discipline quali: la meccanica, l’elettronica, la biologia e l’informatica.

MOE è stato sviluppato nell’ambito del laboratorio di intelligenza artificiale e robotica con lo scopo di creare di un sistema di visione stereoscopico mobile a basso costo in grado di emulare le prestazioni dell’apparato visivo umano con notevole precisione. Nel campo della robotica tale progetto risulta fortemente originale dato che i sistemi di visione normalmente dispongono di una sola telecamera mobile oppure di una coppia di telecamere fisse. La costruzione di MOE può sintetizzarsi in tre parti fondamentali: la creazione della meccanica, lo sviluppo dell’elettronica di controllo e l’implementazione dell’interfaccia software per gestire il tutto.

Nella creazione del sistema meccanico abbiamo valutato i componenti, i motori, i materiali e le diverse possibili disposizioni delle telecamere. Nella progettazione del sistema elettronico che controlla il movimento di MOE abbiamo vagliato l'efficienza dei motori passo passo, l'interazione con i sensori di finecorsa e l'alimentazione di tutto il sistema. La parte software infine si occupa dei comandi da dare al microchip che controlla i motori quali *get* per ricevere la posizione, *set* per comandare lo spostamento in una specifica posizione e *reset* per tornare nella posizione di partenza. Per lo sviluppo di MOE sono state seguite delle specifiche piuttosto stringenti sul peso della struttura, sulla velocità di rotazione delle telecamere e sulla precisione del movimento. Sono state effettuate inoltre delle ricerche sulle proporzioni ed i movimenti degli occhi umani quali l'inseguimento lento e veloce (saccadi). In base alle specifiche è stato progettato lo scheletro meccanico. Il risultato è da considerarsi originale; numerose sono infatti le differenze rispetto a robot esistenti come i-RoK, icub o probot. Tali differenze sono l'utilizzo di motori passo passo invece che servomotori elettrici, l'asimmetria della struttura meccanica costruita e l'uso di una trasmissione diretta tra motori e telecamere al fine di ridurre al massimo eventuali giochi. La parte elettronica è composta concettualmente da due stadi: lo stadio di potenza adibito all'alimentazione dei motori e lo stadio logico per il movimento e il controllo di posizione degli occhi del robot. L'interfaccia software consta di una libreria in c per sistemi operativi linux, e si pone come un layer di basso livello nella progettazione di routine di controllo più complesse. Abbiamo infine effettuato dei test qualitativi sulla struttura finale per valutarne la precisione e le velocità che è possibile raggiungere per verificare se le specifiche fossero soddisfatte o meno. Buoni risultati sono stati ottenuti per quanto riguarda la precisione del movimento; i motori passo passo pilotati in half step e quindi con una risoluzione di 0.9° hanno intrinsecamente un errore di risoluzione massimo di 0.45° , a questo valore sommiamo un errore sistematico medio rilevato di 0.0547° ed otteniamo un errore massimo di mezzo grado che riteniamo pienamente soddisfacente per una struttura costruita artigianalmente. Per quanto riguarda la velocità della rotazione i risultati sono ottimi per spostamenti concentrati lungo l'asse orizzontale ma, le prestazioni iniziano a degradare man mano che cresce la componente verticale del movimento. Questo perché il motore che fornisce il movimento verticale alla struttura contenente le telecamere ed i motori orizzontali si è rivelato sottodimensionato per effettuare movimenti *saccadici* di ampiezza superiore ai 5° , nonostante si comporti bene per un normale inseguimento lento. Il peso dell'intera struttura è risultato leggermente eccessivo a causa dei motori e dei numerosi cavi. Abbiamo svolto delle ricerche su nuovi componenti in grado di

aumentare le prestazioni e diminuire il peso della struttura senza stravolgere la parte elettronica, abbiamo effettivamente trovato un motore passo passo che potrebbe sostituire quello usato attualmente a patto di incrementare leggermente il budget dell'intero sistema. Tale componente è stato incluso nei datasheet presenti nella sezione B. Riteniamo che lo svantaggio maggiore della struttura sia lo spostamento del baricentro verso il motore che aziona il movimento verticale. Questo scostamento potrebbe essere problematico se si volesse abbinare il movimento del collo del robot Maximum One azionato da muscoli di McKibben, in questo caso bisognerebbe controbilanciare la struttura o riposizionare il motore che causa questa asimmetria ad esempio con l'utilizzo di cinghie che però deteriorerebbero la precisione del sistema.

L'elaborato di tesi è strutturato nel modo seguente:

Nella seconda sezione illustriamo lo stato dell'arte nel campo delle teste robotiche in particolare descriviamo i tre robot Probo, Icub e i-RoK che presentano notevoli somiglianze con la tesi in oggetto, successivamente esponiamo due particolari sistemi di attuazione di telecamere: il primo tramite muscoli artificiali ed il secondo tramite una coppia di servomotori.

Nella terza sezione elenchiamo le specifiche richieste in fase di progettazione. Tali specifiche comprendono: un peso non superiore al chilogrammo, un campo visivo di estensione simile a quella umana, una velocità di rotazione delle telecamere in grado di emulare quella umana sia per i movimenti di inseguimento lento ($100^\circ/\text{s}$) sia per i movimenti saccadici ($500^\circ/\text{s}$), una risoluzione ed un frame rate delle telecamere adeguati ed infine un'elevata precisione nel movimento che possiamo tradurre in un errore massimo tollerato pari al mezzo grado.

Nella quarta sezione chiariamo termini quali bionica, biorobotica, biomimetica e robotica bioispirata, quest'ultimo ci darà l'occasione di esporre le caratteristiche che nella nostra tesi sono state appunto ispirate dalla natura dell'occhio umano. Per questo motivo inizieremo col descrivere l'occhio, la sua anatomia ed il suo campo visivo per poi continuare classificando i movimenti oculari quali l'inseguimento lento, veloce e la vergenza. Infine verranno descritte le caratteristiche bioispirate del sistema MOE quali le proporzioni, l'ampiezza del campo visivo e la velocità di rotazione.

Nella quinta sezione descriviamo l'apparato meccanico di MOE, accennando i primi tre prototipi preliminari per poi descrivere i quattro prototipi effettivamente realizzati concentrando l'attenzione sulle parti che ogni prototipo ha migliorato rispetto al precedente.

Nella sesta sezione illustriamo l'apparato elettronico MOE. Elenchiamo inizialmente i componenti integrati utilizzati; successivamente presentiamo

gli schemi elettrici realizzati nel prototipo.

Nella settima sezione descriviamo il firmware per microprocessore PIC realizzato, illustriamo le funzionalità del PIC e discutiamo le scelte tecniche effettuate.

Nella ottava sezione presentiamo la libreria libmoe, portando come caso di studio uno scenario di utilizzo di tale libreria per la realizzazione di un controllo software complesso.

Nella nona sezione descriviamo i test effettuati sul sistema completo per verificarne il funzionamento. I test vertono sul controllo di ogni possibile posizione dei motori e di ogni possibile velocità erogabile al fine di ottenere delle valori limite di corretto funzionamento. Infine enunciamo i test di precisione effettuati ed esponiamo il modo in cui sono stati raccolti i dati tramite l'ausilio del toolbox per matlab Calibox.

Nella decima ed ultima sezione commentiamo il lavoro svolto e traiamo delle conclusioni qualitative, evincendo pregi quali la precisione e basso costo di produzione e i difetti riscontrati quali velocità massime per il movimento orizzontale leggermente inferiori alle specifiche e peso superiore alle specifiche. Esponiamo infine le possibili varianti e miglioramenti futuri che permetterebbero di soddisfare a pieno le specifiche, indicando inoltre il dazio di tali cambiamenti in termini di lavoro, precisione e costi.

Capitolo 2

Stato dell'arte

“C'è vero progresso solo quando i vantaggi di una nuova tecnologia diventano per tutti.”

Henry Ford (1863-1947), industriale americano ...

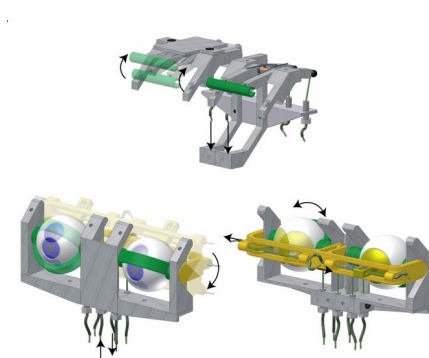
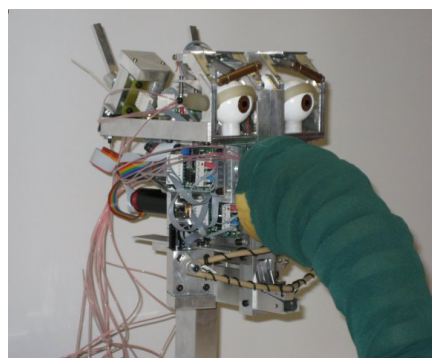
In questo capitolo descriveremo il livello tecnologico raggiunto nell'ambito delle teste robotiche e dei sistemi visivi. Descriveremo nella prima sezione tre diversi robot evincendone le similitudini con il lavoro che abbiamo svolto. Nella seconda sezione invece descriveremo delle possibili tecnologie alternative che potrebbero essere sviluppate in futuro per il nostro sistema.

2.1 Teste robotiche

In questa sezione descriveremo le teste robotiche sviluppate in questi ultimi anni, focalizzando l'attenzione, non tanto sull'interazione uomo-macchina che pare essere il leitmotiv che accomuna ogni tesi di robotica umanoide, quanto sulle somiglianze rispetto al sistema MOE da noi sviluppato.

2.1.1 Probo

Probo [14] [15] è un robot sviluppato nell'ambito dello studio dell'interazione uomo-robot. Il robot è dotato di una testa completamente attuata, dotata di 20 gradi di libertà e capace di esprimere emozioni umane. Come mostrato in figura 2.1, gli occhi del robot sono stati progettati in modo da garantire interazione oculare con l'osservatore, per quanto riguarda l'inseguimento e l'espressione di emozioni tramite l'uso combinato di occhi e sopracciglia. Il robot, inoltre, è provvisto di una proboscide dotata di 3 gradi di libertà attuata da tre motori brushless in maniera analoga al collo del robot E_2

*Figura 2.1: Occhi del robot probo**Figura 2.2: Immagine del robot probo*

sviluppato dal Politecnico di Milano. Rispetto alla tesi sviluppata in questo elaborato, la struttura meccanica di Probo è sicuramente molto più complessa ed improntata al design particolare del robot, ciò nonostante il tipo di movimenti oculari è del tutto simile a quello delle telecamere utilizzate in MOE. Il campo visivo dei proba infatti è di circa 100° orizzontalmente e di circa 80° verticalmente e cioè leggermente più piccolo del campo visivo di MOE; c'è da aggiungere però che Probo dispone di un efficiente collo mobile che permette di allargare ulteriormente il campo visivo totale. A differenza del progetto in esame, il sistema visivo è azionato da servomotori elettrici. Di particolare interesse infine è la struttura dei bulbi oculari mostrati nella figura 2.1 dove si può notare una complessa struttura meccanica di sostegno che presenta la stessa assunzione di MOE per quanto riguarda il movimento verticale, questo infatti, come spiegheremo nel capitolo 5, è comune ad entrambi gli occhi e può essere quindi attuato allo stesso modo per entrambe le camere utilizzate.

2.1.2 Icub

Icub è un robot sviluppato nel 2006 con l'obiettivo di essere il più completo robot umanoide in termine di complessità cinematica con un totale di 53 gradi di libertà. Analizzando nello specifico il sistema visivo, il team di sviluppo di icub ha compiuto diversi approfondimenti sui movimenti oculari al fine di emulare le prestazioni umani esattamente come nel presente elaborato.

Per quanto riguarda il campo visivo, Icub è dotato di un estensione di 90° orizzontalmente e di 80° verticalmente ma, come Probo, è dotato di un collo mobile che ne amplia le dimensioni. Nelle specifiche del robot Icub ci vengono forniti dei dati sulle velocità raggiungibili dagli occhi. Gli occhi di Probo raggiungono velocità di rotazione estremamente elevate, pari a

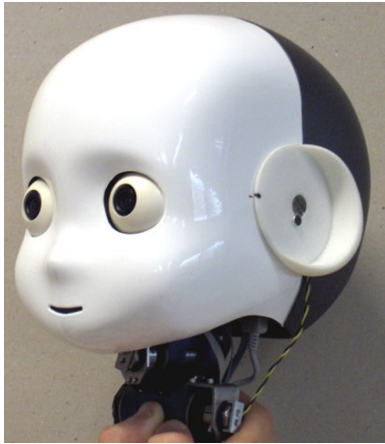


Figura 2.3: Testa robotica di i-cub

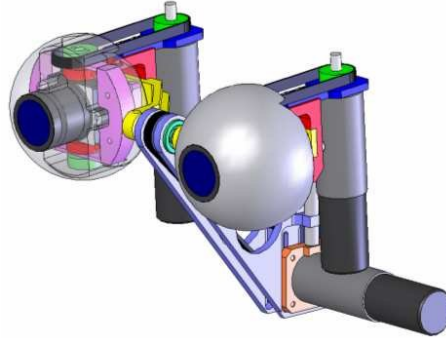


Figura 2.4: Meccanismo utilizzato per gli occhi di i-cub

850°/s come massimo e 166°/s come minimo. Queste velocità superano di gran lunga quelle raggiunte nel sistema MOE e perfino quelle raggiungibili dall'occhio umano. Anche in questo progetto notiamo che il movimento verticale degli occhi è comune ad entrambe le telecamere e che le telecamere sono azionate da servomotori elettrici. Il peso totale della testa di Icub è di 1.5kg, similmente alle specifiche fornite per MOE [22] [23].

2.1.3 i-RoK



Figura 2.5: Testa robotica i-RoK

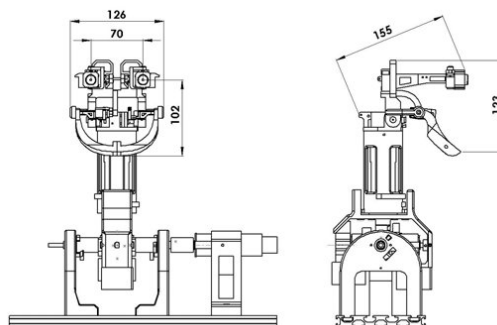


Figura 2.6: Misure di i-RoK in mm

Il progetto i-Rok, come i precedenti, è stato sviluppato per meglio comprendere le interazioni uomo-macchina in un contesto sociale. La testa sviluppata nel 2008 in Turchia è dotata di 8 gradi di libertà che permettono al robot di emulare gli stessi movimenti umani. Una testa robotica antropo-

morfa rappresenta una sfida ardua nel campo della robotica, questo perché esistono numerose caratteristiche da emulare quali:

1. le dimensioni della testa e dei componenti;
2. i numerosi giunti presenti nella testa, collo, occhi, etc;
3. l'estensione dei movimenti;
4. il numero dei gradi di libertà;
5. il peso;
6. la forza dei giunti;
7. i movimenti;

Numerose sono le analogie tra il robot i-RoK e la testa robotica oggetto della presente tesi; prima di tutto le proporzioni quali la distanza tra il centro delle telecamere che risulta per entrambi i robot pari a 7 cm per emulare la distanza tra gli occhi umani. L'estensione del campo visivo di i-Rok è di 90° gradi orizzontalmente e di 40° verticalmente. Un campo visivo così ristretto è probabilmente dettato dalla complessa struttura meccanica che sostiene le telecamere, struttura visibile nella figura 2.7. Anche per la costruzione di i-RoK sono stati effettuati degli studi biologici sui movimenti oculari umani e i risultati sono i medesimi di quelli evinti nel presente elaborato. I-RoK riesce a compiere l'inseguimento lento di un oggetto a circa $100^\circ/\text{s}$ e se necessario, riesce a compiere movimenti saccadici a $500^\circ/\text{s}$. Le prestazioni finali del robot sono quindi molto buone; purtroppo nessun dato ci viene fornito riguardo alla precisione del sistema così come negli altri progetti di teste robotiche. Concludiamo la presentazione di I-Rok ricordando che anche in questo caso sono stati utilizzati dei servomotori sia per i movimenti di pan e tilt delle camere che per l'attuazione del collo. In particolare il movimento verticale degli occhi è fornito tramite una cinghia ed è comune ad entrambi gli occhi [20].

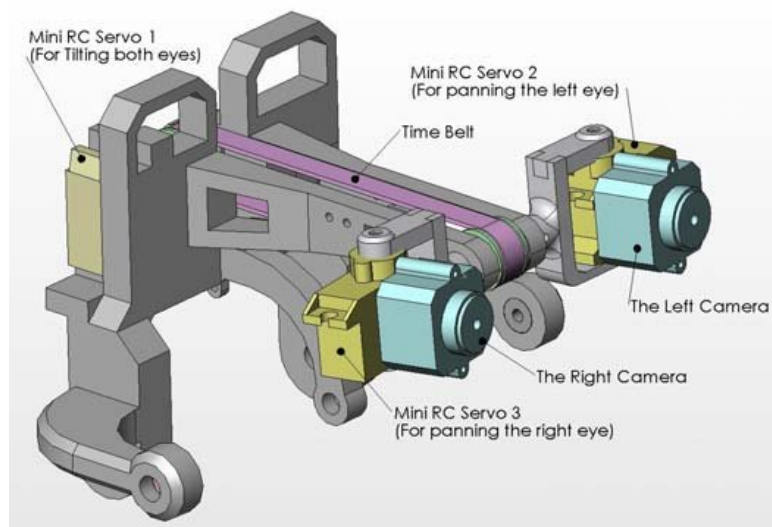


Figura 2.7: Struttura che permette il movimento oculare di i-RoK

2.2 Sistemi visivi

2.2.1 Telecamere attuate tramite PAM

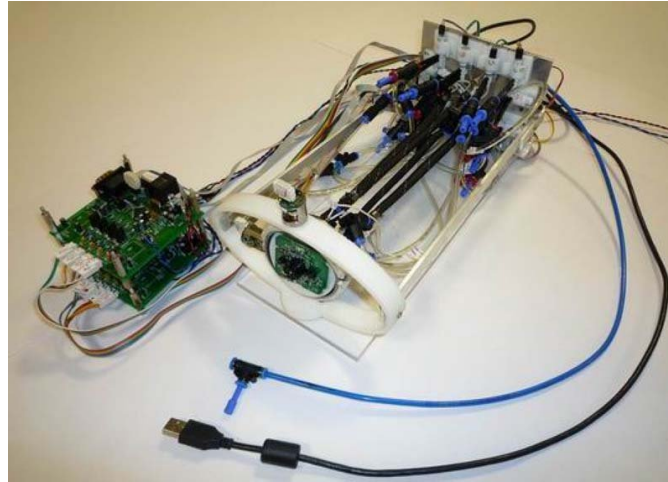


Figura 2.8: Telecamera Attuata da muscoli artificiali

Un recente progetto del 2009 indagava la possibilità di attuare una telecamera tramite muscoli artificiali. Lo studio investigava il problema del controllo degli attuatori e di stabilizzazione delle immagini in relazione ai disturbi rilevati. Un robot attuato da muscoli pneumatici artificiali presenta infatti numerose difficoltà nel controllo rispetto ad attuatori elettrici necessitando un controllo PWM¹ delle valvole che regolano la pressione dell'aria.

Il vantaggio di usare attuatori pneumatici è senza dubbio la duttilità dei materiali con i quali sono costruiti, ciò rende possibile realizzare dei *soft robot* che non presentano una struttura rigida come succede nei robot attuati elettricamente o idraulicamente. L'utilizzo di una struttura deformabile è motivata dalla necessità di minimizzare le conseguenze di una possibile collisione tra un essere umano ed il robot. L'ambito in cui una tecnologia del genere può essere applicato è la cura dei disabili o di individui anziani; tale obiettivo sta diventando uno degli aspetti più importanti nella ricerca robotica. Conosciuti anche con il nome di muscoli artificiali di McKibben [19] [25], questi attuatori, nel corso degli anni, sono stati migliorati soprattutto dal punto di vista dei materiali e sono ora conosciuti come PAM². Un PAM

¹PulseWidth Modulation

²Pneumatic Artificial Muscles

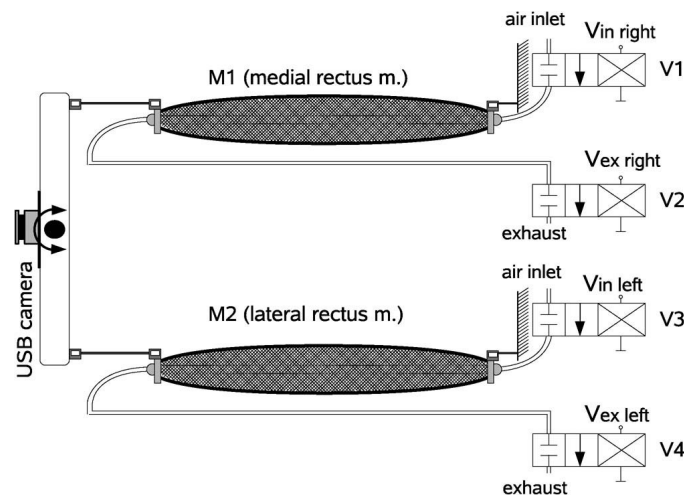


Figura 2.9: Sistema di PAM antagonisti

è costituito da una parte cilindrica cava in gomma, che si allunga quando è riempita di aria e si contrae quando ne viene svuotata. Come i muscoli umani i PAM operano a coppie antagoniste dato che riescono ad esercitare forza solamente durante la fase di contrazione. Le proprietà dinamiche di un PAM sono non-lineari ma si possono assumere come lineari sotto certe appropriate condizioni, come nel caso di piccole contrazioni concentrate nella fase centrale del movimento [3].

2.2.2 Sistemi Pan-Tilt

Un sistema motorio non antropomorfo ma di una certa valenza sperimentale è il sistema pan&tilt. Nel progetto Headroid1 [30] una telecamera del genere è stata utilizzata con un algoritmo di riconoscimento facciale per muoversi in modo da tenere l'interlocutore al centro del campo visivo. Un sistema di questo tipo montato su di una testa robotica sarebbe scarsamente antropomorfo ma potrebbe rivelarsi molto semplice da costruire e leggero dato che non necessita di grosse strutture di sostegno per i servomotori. L'algoritmo di riconoscimento facciale utilizzato per Headroid inoltre potrebbe essere utilizzato in un futuro anche per il sistema MOE.



Figura 2.10: Sistema pan and tilt Headroid

Capitolo 3

Specifiche richieste

“Non si desidera ciò che è facile ottenere.”

Ovidio (43 a.C.-17 d.C.), poeta latino ...

In questa sezione descriveremo le specifiche richieste per il completamento della tesi. Lo studio di fattibilità ha sostanzialmente riguardato il sistema di visione ed il peso. In particolare dovevano essere soddisfatti determinati vincoli quali: peso massimo di circa un chilogrammo, campo visivo simile a quello umano, proporzioni del sistema visivo umano, velocità di rotazione delle telecamere pari a quella umana sia nel caso di movimenti di inseguimento lento che di inseguimento veloce (saccadi) e telecamere con una risoluzione e frame rate sufficiente per eseguire object tracking.

3.1 Specifiche bioispirate

Le prime specifiche sorgono dalla volontà di creare un sistema bioispirato, per questo motivo per le prime tre specifiche l'obiettivo sarà quello di avvicinarsi il più possibile al sistema visivo umano.

3.1.1 Campo visivo

Come descriveremo nel capitolo 4, il campo visivo umano è fortemente asimmetrico; l'occhio infatti può scorgere oggetti fino a 60° internamente, 95° esternamente, 75° verso il basso e 60° verso l'alto. L'estensione totale che abbiamo preso come riferimento per il nostro sistema è quindi di 155° orizzontalmente e di circa 135° verticalmente per il singolo occhio [1].

3.1.2 Proporzioni

Al fine di emulare le proporzioni dell'apparato visivo umano le telecamere sono state disposte in modo da ottenere una distanza di 7 cm tra i due centri degli obiettivi. Le telecamere stesse sono state scelte di forma sferica in modo da ricordare il bulbo oculare umano.

3.1.3 Velocità di rotazione

La velocità di rotazione delle telecamere doveva essere pari a quella umana sia per i movimenti di inseguimento lento pari a circa $100^\circ/\text{sec}$ sia per i movimenti di inseguimento veloce (saccadi) pari a circa $500^\circ/\text{s}$. Tali specifiche riguardano sia le rotazioni di pan che di tilt degli occhi.

3.2 Specifiche non bioispirate

Abbiamo dovuto tenere conto inoltre di specifiche non bioispirate al fine di progettare una struttura in grado di funzionare correttamente, una volta inserita sul robot Maximum One, per il completamento di task specifici quali l'inseguimento di oggetti o riconoscimento facciale.

3.2.1 Risoluzione e frame rate

La risoluzione delle telecamere doveva essere di almeno 640×480 , con un Frame rate di 24-30 Frame/s. I driver per il loro funzionamento dovevano essere compatibili con linux. Le telecamere inoltre dovevano avere un basso costo ed essere dotate di uscita usb ed il loro peso non doveva superare i 100 grammi; per questo motivo sono state scelte le webcam logitech c200.

3.2.2 Elevata precisione

La precisione degli spostamenti doveva essere molto elevata. Erano tollerati errori massimi di 0.5° globali indipendentemente dalla posizione richiesta. Nel caso di motori passo passo l'errore globale sarà dato dalla metà del passo del motore a cui sommiamo l'errore di sistema.

3.2.3 Peso

Come mostrato nella figura 3.1 il robot Maximum One monta un collo mobile azionato da attuatori di McKibben, il peso massimo che può sostenere per riuscire a muoversi in maniera adeguata è stato calcolato attorno al chilogrammo. Un peso ridotto dell'ordine di grandezza di questo tipo è una

specifica importante nel caso si volesse utilizzare il sistema MOE direttamente collegato al robot Maximum One. Dobbiamo ricordare che test sul sistema collo-testa non hanno potuto aver luogo, per questo motivo ci auguriamo che possa essere un possibile sviluppo futuro del progetto.

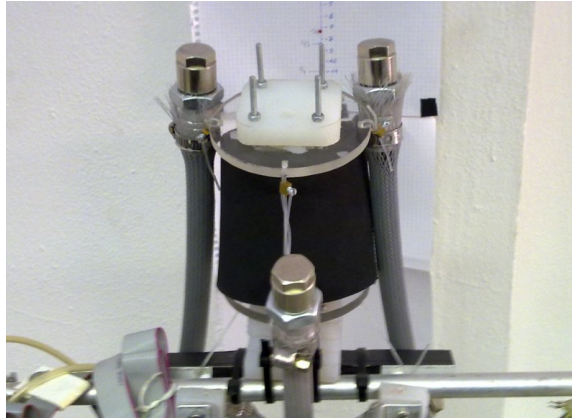


Figura 3.1: Primo piano del collo del robot Maximum One

Capitolo 4

Bioispirazione

“La conoscenza di ciò che è non apre direttamente la porta alla conoscenza di ciò che dovrebbe essere. Si può avere la conoscenza più chiara e più completa di ciò che è, e tuttavia non riuscire a dedurre da questa quale dovrebbe essere la meta delle nostre aspirazioni umane. La conoscenza obiettiva ci fornisce strumenti potenti per la conquista di certe mete, ma il fine ultimo e il desiderio di raggiungerlo devono nascere da un'altra fonte. ”

Albert Einstein (1879-1955), fisico tedesco ...

In questa sezione definiremo i termini biorobotica e robotica bioispirata e successivamente descriveremo l'occhio umano, i suoi muscoli, la precisione della visione umana e l'estensione del campo visivo. Classificheremo e descriveremo inoltre i movimenti oculari quali l'inseguimento lento, saccadi e vergenza ed infine descriveremo quali caratteristiche dell'occhio umano sono state riprese per lo sviluppo del sistema MOE.

4.1 Evoluzione della robotica e biorobotica

Il progresso tecnologico e l'avvento della microelettronica hanno portato alla realizzazione di macchine robotiche sempre più avanzate che integrano strutture e meccanismi, attuatori e sensori, elettronica di processo e di controllo, sorgente d'energia ed interfaccia persona-macchina. La definizione di robot stessa quindi si è evoluta nel corso degli anni e, ad oggi, la definizione che meglio riflette la situazione attuale è probabilmente la seguente: *un robot è una macchina mecatronica in grado di generare e controllare movimenti e forze*. Secondo questa moderna concezione un robot è una macchina che percepisce, pensa e agisce, inserita nel mondo reale con il quale interagisce. Un robot possiede un corpo dotato di massa, soggetto a tutte le leggi fisiche

che regolano il mondo circostante, e grazie all'elettronica integrata è dotato di un certo grado di intelligenza che gli consente di svolgere alcuni task in modo autonomo [13]. L'analogia fra un robot definito in questo modo e un sistema biologico è evidente: entrambi sono dotati di un sistema di locomozione, sensori ed attuatori, richiedono un sistema di controllo, interagiscono con l'ambiente. I sistemi viventi sono la dimostrazione che molte funzioni complesse, come ad esempio la locomozione, la visione o la manipolazione, che i ricercatori in robotica sognano di vedere attuate, possono davvero essere realizzate.

I sistemi biologici quindi non sono più soltanto i principali destinatari dei servizi dei sistemi robotici ma, possono anche diventare un'importante fonte di ispirazione per componenti e comportamenti dei robot del futuro [21].

La profonda sinergia venuta a crearsi tra la robotica e gli organismi biologici ha portato alla nascita di una nuova disciplina: la biorobotica.

4.1.1 Bionica, biorobotica, biomimetica e robotica bioispirata

Il termine bionica fu coniato nel 1960 dall'ingegnere Jack Steel con il significato di *scienza dei sistemi con caratteristiche copiate dalla natura*. Nel corso degli anni il termine bionico acquisì una differente connotazione e venne associato sempre più allo sviluppo di arti e di corpi artificiali dotati di forza sovrumana [2].

Successivamente Otto Schmitt, un inventore americano, coniò il termine biomimetica per descrivere il trasferimento di idee dalla biologia alla tecnologia. Ne 1974 venne ufficialmente definito il termine biomimetica come *lo studio consapevole dei processi biologici e biomeccanici della natura, come fonte di ispirazione per il miglioramento delle attività e tecnologie umane*.

Recentemente è stata definita biorobotica una nuova area scientifico / tecnologica che fonde robotica e bioingegneria; in particolare la biorobotica è la scienza e la tecnologia della progettazione e della realizzazione di sistemi robotici di ispirazione biologica o di applicazione biomedica [9].

La biorobotica può essere intesa e studiata in due diverse prospettive:

Robotica biomimetica come scienza che serve a generare nuove scoperte e quindi nuova conoscenza, contribuendo così al progresso scientifico;

Robotica bioispirata come ingegneria per inventare e generare nuova tecnologia;

La robotica bioispirata ha quindi l'obiettivo di approfondire le conoscenze sul funzionamento dei sistemi biologici da un punto di vista ingegneristico e

biomeccatronico e di utilizzare tali migliori conoscenze al fine di sviluppare metodologie e tecnologie innovative sia per la progettazione che per la realizzazione di nuove macchine e sistemi.

Nella prossima sezione descriveremo gli aspetti biologici che hanno influenzato la costruzione del sistema MOE.

4.2 Caratteristiche visive umane

Il sistema visivo umano è composto da 3 parti : l'occhio, la retina ed il cervello [29]. L'occhio è il sistema ottico che forma e proietta le immagini su di una superficie sensibile, tale superficie è chiamata retina ed ha diversi compiti quali: raccogliere le diverse informazioni riguardanti le immagini, eseguire una prima elaborazione di tali immagini e trasferire l'informazione ai centri superiori. L'immagine definitiva viene ricomposta dal cervello che elabora i dati raccolti dalla retina. Descriveremo brevemente nella prossima sezione la struttura dell'occhio umano per comprendere quali caratteristiche sono state di ispirazione per il nostro sistema.

4.2.1 L'occhio umano

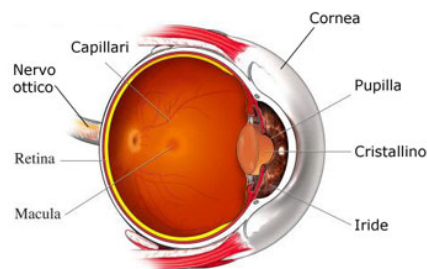


Figura 4.1: La struttura dell'occhio umano

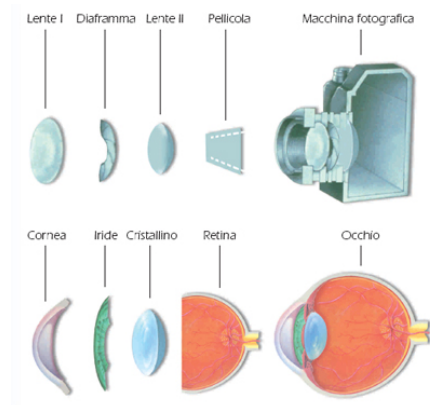


Figura 4.2: L'occhio come una macchina fotografica

L'occhio umano è l'organo di senso principale dell'apparato visivo. L'occhio riceve la luce dall'esterno che, passando attraverso una serie di lenti naturali quali cornea, cristallino e corpo vitreo, arriva fino ad una superficie sensibile la quale è in grado di trasformare l'eccitazione luminosa in segnali elettrici per il nervo ottico. L'occhio umano come una macchina fotografica

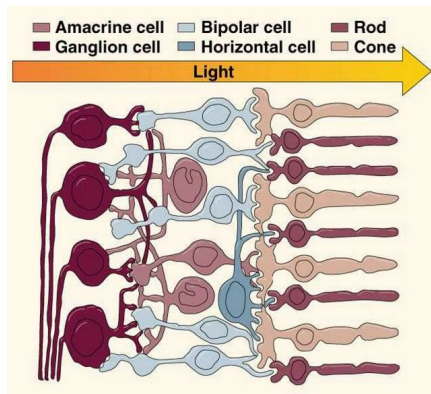


Figura 4.3: L'anatomia della retina

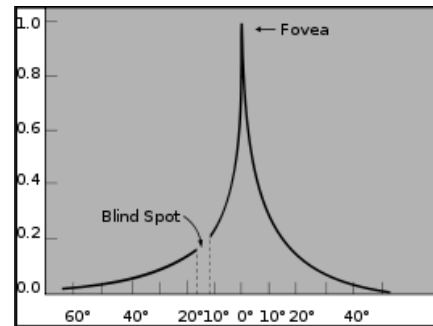


Figura 4.4: Rapporto gangli / fotorecettori

è in grado di imprimere la luce su di una superficie sensibile chiamata retina [6] [7]. La retina è la membrana più interna dell'occhio, essa è composta di cellule sensibili alle radiazioni luminose chiamate fotorecettori. Esistono due tipi differenti di fotorecettori nella retina:

i coni : sensibili sia alle forme che ai colori. Negli esseri umani se ne differenziano tre tipi: sensibili al rosso, al verde e al blu; se attivati simultaneamente la luce percepita risulta essere bianca. Consentono di percepire dettagli e cambiamenti d'immagine più rapidi rispetto ai bastoncelli, poiché i loro tempi di risposta agli stimoli sono più veloci. In ogni occhio il loro numero può variare dai sei ai sette milioni [10]. I coni sono localizzati principalmente nella fovea¹ e permettono la cosiddetta visione *fotopica* come si può notare infatti dall'immagine 4.4 il rapporto tra gangli e fotorecettori è prossimo ad uno al centro della fovea, qui infatti ogni fotorecettore ha un ganglio in grado di ricevere il proprio segnale [12].

i bastoncelli : hanno una funzione complementare rispetto ai coni, essi infatti permettono la visione in condizioni di scarsa luminosità garantendo la cosiddetta visione *scotopica*² ma non la percezione cromatica. I bastoncelli sono abbondanti nella periferia retinica. In ogni occhio il loro numero può variare dai 75 ai 150 milioni [11].

¹parte centrale della retina dove quindi è concentrata l'acuità visiva umana

²detta anche notturna

4.2.2 I muscoli dell'occhio

L'apparato muscolare dell'occhio è collegato alla sclera³, tali muscoli sono in tutto sei, 2 obliqui (inferiore e superiore) e 4 retti (inferiore, laterale mediale, superiore). Gli altri muscoli dell'occhio sono: orbicolare dell'occhio, elevatore della palpebra superiore, ciliare, dilatatore dell'iride e sfintere dell'iride. I muscoli che fanno muovere l'occhio sono organizzati in tre coppie

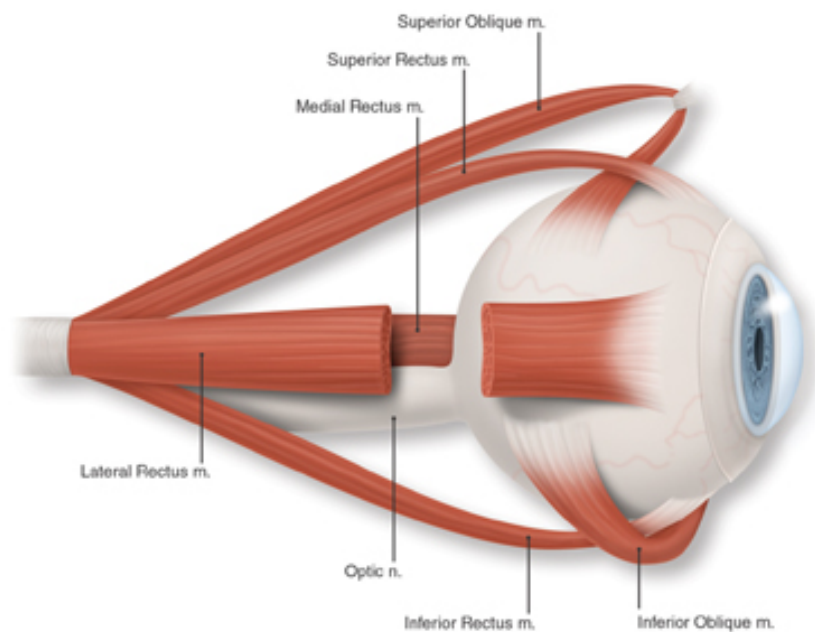


Figura 4.5: I muscoli dell'occhio

ad azione antagonista che fanno ruotare l'occhio sui tre piani dello spazio. Per i movimenti orizzontali (abduzione, in direzione temporale ed adduzione in direzione nasale) intervengono rispettivamente i muscoli retti mediali e laterali. Per i movimenti verticali (elevazione ed abbassamento) e per quelli di torsione sono primariamente coinvolti i muscoli retti superiore ed inferiore e gli obliqui inferiore e superiore. Abbiamo studiato la muscolatura umana per capire quali movimenti eseguisse l'occhio oltre per valutare eventuali sviluppi futuri di una testa robotica con delle telecamere mosse da muscoli artificiali (sezione 2.2.1).

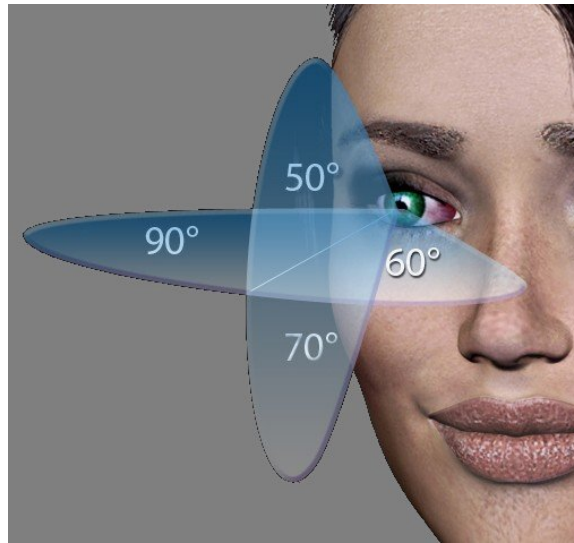


Figura 4.6: Campo visivo umano

4.2.3 Visione e campo visivo umano

Il campo visivo si può definire come la porzione di spazio che un occhio immobile percepisce di fronte a sé. Nell'uomo il campo visivo totale orizzontale coperto dai due occhi si estende per poco più di 180° , questo includendo tutta la visione periferica compresa la cosiddetta coda dell'occhio. Siccome l'ambito di mobilità dell'occhio è di circa 90° , muovendosi gli occhi, la visione panoramica arriva fino a 270° . Accoppiando inoltre la rotazione della testa si riescono a coprire più di 360° . Limitandoci al solo movimento dei muscoli oculari possiamo definire l'estensione del campo visivo per il singolo occhio pari a 60° nasalmente, 50° superiormente, 90° temporalmente e circa 70° inferiormente, assumendo così una sorta di forma ovoidale come mostrato nella figura 4.6.

4.3 Classificazione dei movimenti oculari

La bilaterale integrazione delle attività oculomotorie fa sì che i due occhi costituiscano una singola unità funzionale e realizzino insieme una sorta di occhio ciclopico, capace di provvedere alla visione unica e stereoscopica degli oggetti. I movimenti dei due occhi possono essere uguali (orizzontali,

³il cosiddetto bianco dell'occhio

verticali, di torsione sull'asse antero-posteriore) od opposti (convergenza, divergenza) e soddisfano molteplici esigenze, quali:

1. mantenere stabile l'asse visivo;
2. rintracciare, inseguire e fissare gli oggetti che entrano nel campo visivo;
3. consentire una visione unica e stereoscopica;
4. permettere un'esplorazione attenta dell'ambiente [4];

Di particolare rilevanza per lo sviluppo del sistema MOE sono i movimenti di inseguimento lento, le saccadi e la vergenza.

4.3.1 Inseguimento lento

Il movimento di inseguimento lento consente di mantenere in posizione foveale piccoli bersagli che si spostano sullo sfondo. Solitamente quando si segue un bersaglio mobile, saccadi di piccola ampiezza sono intercalate a periodi di inseguimento lento. Il movimento di inseguimento è efficace a velocità dello stimolo fino a $100^\circ/\text{s}$ e la sua latenza è di circa 100 ms. Inoltre questo movimento non può essere eseguito senza un bersaglio, in tal caso sarebbe una semplice sequenza di saccadi. Questo movimento si avvale di una componente *predittiva* che consente di emettere una risposta motoria non solo senza ritardo allo stimolo ma spesso in anticipo.

4.3.2 Inseguimento veloce - Saccadi

Le saccadi consistono in rapidi movimenti degli occhi eseguiti per portare la zona di interesse a coincidere con la fovea [26]. Vengono eseguite in media 3-4 saccadi al secondo e solitamente possono avere un'ampiezza compresa fra meno di 0.1° ⁴ e quasi 90° ; raramente tuttavia l'ampiezza delle saccadi eccede i 20° dato che in caso di angoli superiori entra in gioco la rotazione del collo e del busto. Non esiste una velocità angolare massima precisa per un movimento saccadico, questo perché la velocità è in funzione dello spostamento effettuato. Per calcolare la velocità massima di una saccade dobbiamo quindi definire l'entità dello spostamento a cui fare riferimento; ad esempio per spostamenti di 10° di ampiezza è associata una velocità di circa $300^\circ/\text{s}$, per 30° una velocità di circa $500^\circ/\text{s}$. La latenza di una saccade è, in media, di circa 225 ms⁵ [17] [16].

⁴microsaccadi

⁵tra la comparsa del bersaglio e l'inizio della saccade passano circa 225 ms

4.3.3 Vergenza

I movimenti analizzati fino ad ora sono detti coniugati perché il movimento dei due occhi è il medesimo. In altri casi, ad esempio nel movimento di vergenza, i movimenti sono disgiunti. I movimenti di vergenza fanno convergere o divergere gli occhi sul piano orizzontale in modo da garantire la fusione binoculare delle immagini. Si accompagnano quindi ad ogni cambio del piano focale. Tradizionalmente si considera il movimento di vergenza caratterizzato da una latenza abbastanza elevata (150-200ms), da una bassa velocità ($15^\circ/\text{s}$) e da un tempo relativamente elevato per raggiungere la velocità massima (circa 1 s).

4.4 Caratteristiche visive emulate nel sistema MOE

4.4.1 Proporzioni

Dato che la distanza media tra i centri focali dei due occhi umani è di circa 7 cm, abbiamo disposto le telecamere alla stessa distanza nella costruzione di tutti i prototipi meccanici di MOE.

4.4.2 Campo visivo

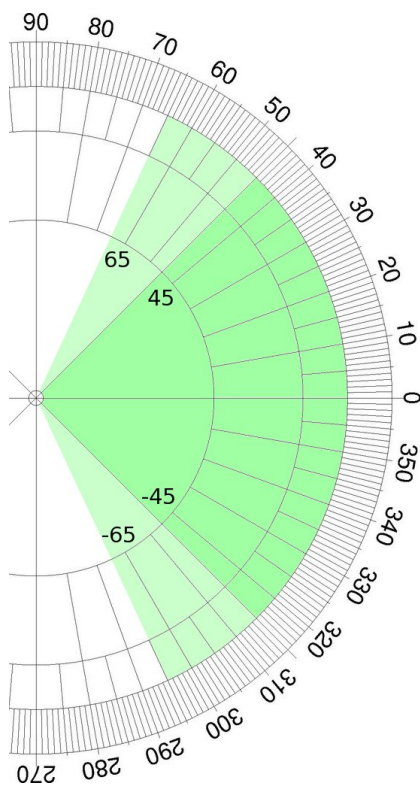


Figura 4.7: Estensione orizzontale del campo visivo di MOE

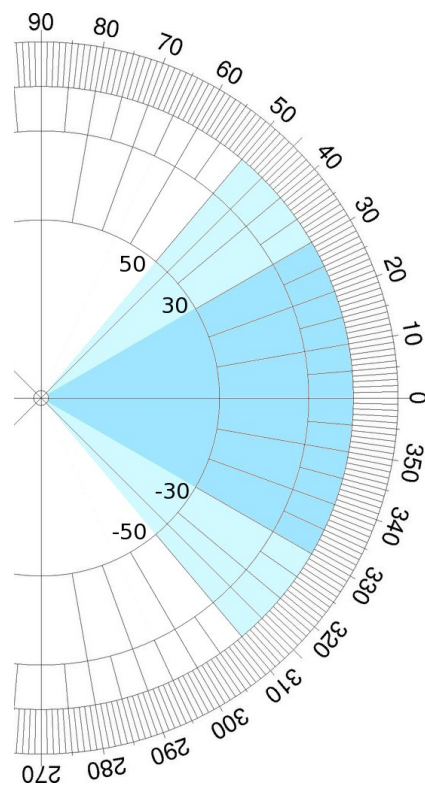


Figura 4.8: Estensione verticale del campo visivo di MOE

Come spiegato nella sezione 4.2.3 il campo visivo umano è fortemente asimmetrico. Per semplificare il riconoscimento degli estremi del campo visivo di MOE abbiamo deciso di renderlo simmetrico rispetto agli assi verticale e orizzontale. In accordo con i progetti studiati nella sezione 2, abbiamo deciso di ridurre l'estensione orizzontale e verticale del campo visivo dato che i movimenti di inseguimento veloce si concentrano fortemente attorno al centro di quest'area. Nell'uomo l'acuità visiva stessa è concentrata nella zona foveale e diminuisce progressivamente nella zona periferica della retina.

Per questi motivi abbiamo ritenuto ragionevole ridurre l'estensione delle rotazioni delle telecamere fino a 90° orizzontalmente e 60° verticalmente. Possiamo aggiungere infine che, nonostante tali dati sembrino sottostimati, essi si basano solamente sullo scostamento del centro dell'immagine a cui poi vanno aggiunti i gradi percepiti dalla telecamera come laterali che possiamo stimare attorno ai 20° come viene mostrato nelle figure 4.7 e 4.8. Dopo questa considerazione l'estensione effettiva della singola telecamera raggiunge i 130° orizzontalmente e 100° verticalmente, estensione che abbiamo ritenuto essere soddisfacente per eseguire i compiti richiesti.

4.4.3 Velocità di rotazione

La velocità di rotazione rilevata con i motori attuali è di circa $665^\circ/\text{s}$ per i movimenti lungo l'asse orizzontale e di $214^\circ/\text{s}$ per i movimenti lungo l'asse verticale. Per emulare l'inseguimento lento umano abbiamo ridotto la velocità a circa $100^\circ/\text{s}$. Per quanto riguarda il movimento saccadico invece abbiamo utilizzato la massima velocità raggiungibile dai motori. Tale velocità massima dipende fortemente dalle componenti che entrano in gioco nel movimento; in particolare essendo il motore adibito al movimento verticale più lento, la velocità massima raggiungibile sarà minore all'aumentare della componente verticale. Per comparare il nostro sistema, inoltre, dobbiamo prendere come riferimento l'entità dello spostamento e non solo la direzione di quest'ultimo rispetto all'orizzonte; ricordiamo infatti che nella sezione 4.3.2 abbiamo quantificato la velocità di saccadi di 10° e 30° di ampiezza pari rispettivamente a $300^\circ/\text{s}$ e $500^\circ/\text{s}$. Possiamo quindi affermare che il sistema emula correttamente il movimento dell'occhio umano nei seguenti casi⁶:

saccadi di 5° di ampiezza sempre;

saccadi di 10° di ampiezza con spostamenti che formano angoli minori di 80° rispetto all'orizzonte (quindi per movimenti non fortemente concentrati lungo l'asse verticale);

saccadi di 30° di ampiezza con spostamenti che formano angoli minori di 65° rispetto all'orizzonte (quindi per movimenti concentrati sull'asse orizzontale).

Come verrà esposto nel capitolo 10 se tali risultati non dovessero essere sufficienti, è possibile aumentare le prestazioni dell'intero sistema sostituendo

⁶estratti dalla tabella 9.1 presente nella sezione 9 a pagina 91

solamente il motore adibito al movimento verticale con il motore proposto nei datasheet finali.

Capitolo 5

Sistema meccanico MOE

“Tutti sanno che una cosa è impossibile da realizzare, finché arriva uno sprovveduto che non lo sa e la inventa.”

Albert Einstein (1879-1955), fisico tedesco ...

In questa sezione spiegheremo come è stato affrontato il problema di creare una struttura meccanica in grado di muovere le telecamere in modo da rispettare le specifiche date. Il percorso che ci ha portato a completare la struttura finale passa attraverso numerosi prototipi che verranno descritti spiegando il motivo per il quale sono stati scartati.

5.1 Prototipi preliminari

5.1.1 Prototipo Nr.1

La necessità di muovere le due telecamere ci ha portato inizialmente a ragionare sulla fattibilità di una coppia di giunti cardanici che potessero al contempo sostenere le telecamere e permettere loro il movimento orizzontale e verticale. In questo primo progetto preliminare pensavamo di posizionare quattro motori passo passo nella parte posteriore della testa e successivamente collegarli alle telecamere tramite cavi inestensibili per trasmettere la rotazione. L'intera struttura andava poi montata su di una base in plexiglas, collegabile al collo del robot Maximum One tramite una struttura in alluminio. Tale ipotesi è stata subito scartata sia per l'impossibilità di posizionare il centro degli assi del giunto cardanico al centro delle telecamere senza smontare completamente le telecamere stesse sia per la difficoltà nel costruire in laboratorio un giunto cardanico senza giochi che avrebbero inevitabilmente reso il sistema impreciso.

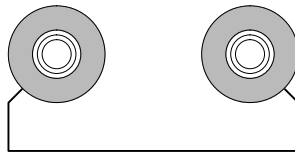


Figura 5.1: Rappresentazione frontale semplificata del prototipo nr.1

5.1.2 Prototipo Nr.2

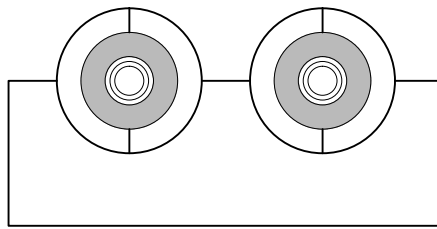


Figura 5.2: Rappresentazione frontale semplificata del prototipo nr.2

Nel secondo prototipo abbiamo cercato di sostituire il giunto cardanico con una struttura meccanica che *avvolgesse* le telecamere e permettesse le rotazioni dall'esterno. Per questo motivo abbiamo ideato una struttura basata su di una circonferenza concentrica rispetto all'obiettivo di ciascuna telecamera che ricordiamo essere di forma sferica. La circonferenza doveva essere in grado di ruotare dall'alto in basso per mezzo dei giunti laterali che ne sostenevano la struttura. All'interno della circonferenza avevamo previsto di posizionare un asse verticale che, passando all'interno del bulbo della telecamera, permettesse la rotazione destra/sinistra. Tali circonferenze dovevano essere movimentate ancora una volta da cavi inestensibili collegati ai motori. L'impossibilità, in laboratorio, di creare cerchi con sufficiente precisione ci ha fatto scartare questa seconda configurazione. Abbiamo inoltre iniziato a ragionare sul movimento verticale cercando di documentarci se questo fosse effettivamente accoppiato per entrambi gli occhi.

5.1.3 Prototipo Nr.3

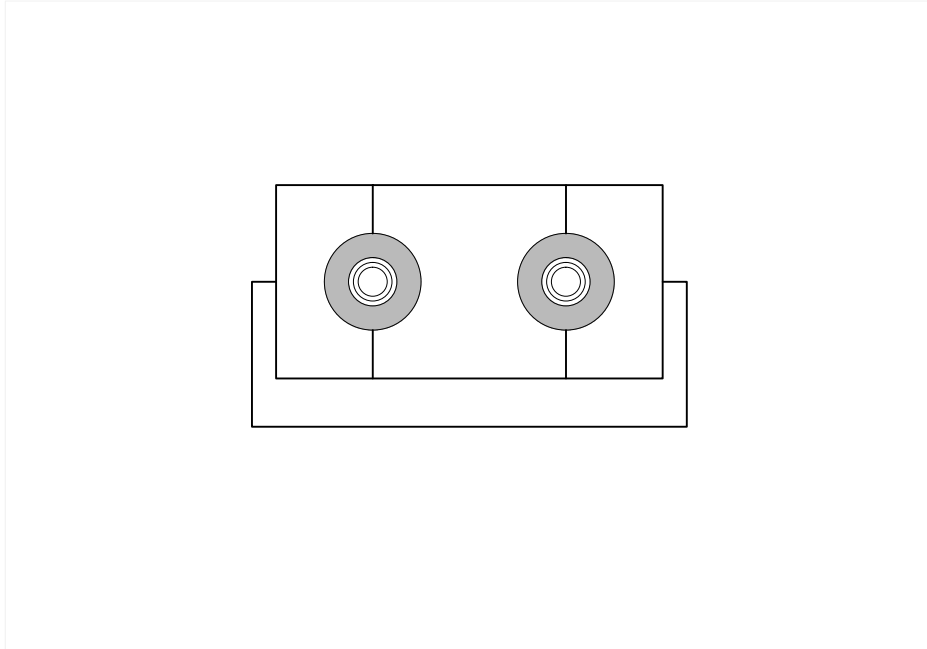


Figura 5.3: Rappresentazione frontale semplificata del prototipo nr.3

Nel terzo prototipo abbiamo deciso di sostituire la struttura cilindrica che doveva sostenere le telecamere con una simile ma di forma rettangolare; in questo modo la creazione in laboratorio è stata notevolmente semplificata. Abbiamo inoltre deciso di ridurre il numero di motori da quattro a tre, dopo aver verificato che il movimento orizzontale degli occhi umani fosse accoppiato. Per questo motivo abbiamo deciso di utilizzare un solo motore per fornire il movimento verticale alla struttura di sostegno per le telecamere. I vantaggi di questa struttura erano la semplicità di costruzione, la relativa leggerezza della struttura contenente le telecamere e quindi la possibilità di utilizzare motori con prestazioni inferiori e costi modesti.

5.2 Prototipi costruiti

Dopo alcuni progetti preliminari, abbiamo iniziato a lavorare sulle strutture fisiche dei prototipi per acquisire dimestichezza con i materiali usati quali plexyglas e alluminio e le macchine utensili presenti in laboratorio.

5.2.1 Prototipo Nr.4

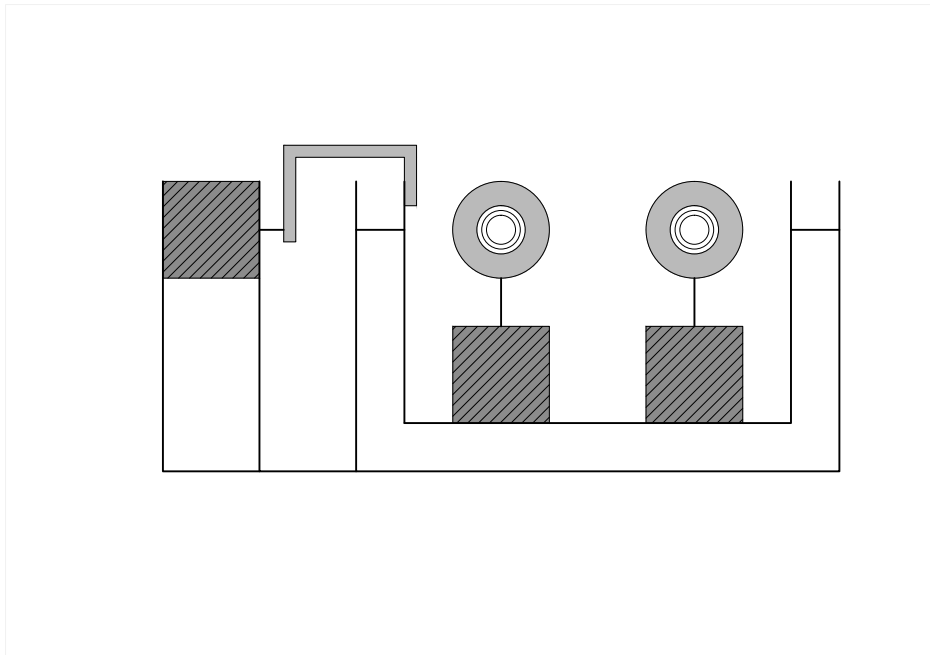


Figura 5.4: Rappresentazione frontale semplificata del prototipo nr.4

L'idea che accomunava i primi tre progetti era l'utilizzo di cavi inestensibili per trasferire a distanza il movimento dalla parte posteriore della testa alla parte anteriore che conteneva le telecamere. Sebbene l'idea garantisse un ottimo bilanciamento della testa, l'utilizzo dei suddetti cavi è stato scartato appena è iniziato il montaggio dei componenti; questo perché una struttura a cavi presentava numerosi problemi di giochi per quanto riguarda gli attacchi dei cavi, sia dalla parte del motore sia dalla parte delle telecamere. Vi era inoltre il problema della scelta del cavo robusto abbastanza da non allungarsi sotto tensione ed infine il problema di posizionare tutti i cavi in tensione e quindi di assicurare che la struttura fosse sufficientemente rigida da sopportare tale tensione anche in movimento. Per questi motivi il primo prototipo reale presentava una trasmissione diretta tra motore e carico. Il

carico consisteva in una struttura a U su cui erano fissati i motori sopra i quali erano a loro volta fissate le telecamere con colla a caldo. La struttura era sorretta a destra e sinistra da placche di alluminio. Per facilitare la rotazione della suddetta parte mobile, abbiamo montato quattro cuscinetti a sfera al cui interno far passare un albero del diametro di 0.5 cm e di lunghezza pari a 3 cm sia per la parte destra che per la parte sinistra. A sinistra della struttura, il motore era montato su due placchette di lunghezza simili a quelle che sostenevano la struttura contenente le telecamere, in questo modo l'albero del motore aveva un asse che coincideva con quello dell'albero che permetteva la rotazione della struttura a U. La trasmissione della rotazione del motore era fornita da una struttura esterna che poggiava sulla parte mobile contenente i motori. Le due telecamere erano posizionate in modo che tra i centri dei due obiettivi ci fossero esattamente 7 cm come da specifiche. I sensori di fine corsa erano montati direttamente sotto le telecamere come è mostrato nella figura 5.5. Una placchetta in alluminio fissata sull'albero del motore attivava il sensore presente nella parte posteriore dei motori.



Figura 5.5: Parte bassa della struttura mobile del prototipo nr.4.

5.2.2 Prototipo Nr.5

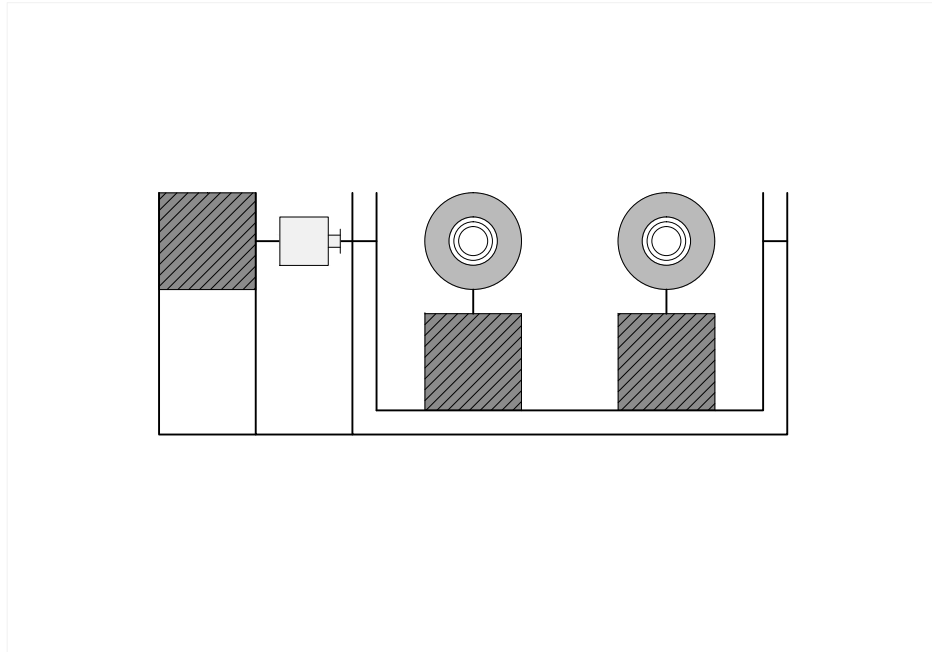


Figura 5.6: Rappresentazione frontale semplificata del prototipo nr.5

Abbiamo notato che il prototipo Nr. 4 non riusciva a muovere la struttura contenente i due motori per il movimento orizzontale, questo a causa dell'eccessivo peso della struttura mobile, della insufficiente potenza del motore adibito al movimento verticale e per l'eccessiva leva tra il carico della parte mobile e l'asse di rotazione. Abbiamo quindi ridotto al minimo tale leva abbassando l'attacco delle telecamere direttamente sull'albero dei motorie rimuovendo temporaneamente i sensori di fine corsa. Al fine di fornire un aiuto al motore verticale abbiamo aggiunto un riduttore 2:1 per verificare se in questo modo fosse possibile muovere la struttura. Abbiamo inserito il riduttore tra carico e motore e lo abbiamo fissato direttamente ad entrambi gli estremi. Purtroppo nonostante il riduttore e la leva ridotta, la parte mobile ancora non riusciva a ruotare per più di qualche grado.

5.2.3 Prototipo Nr.6

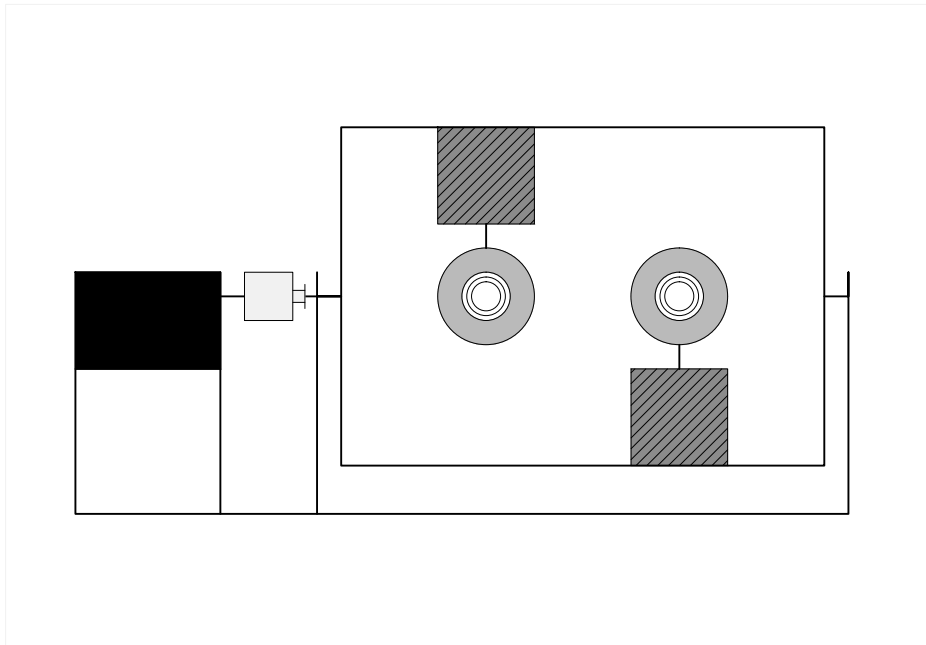


Figura 5.7: Rappresentazione frontale semplificata del prototipo nr.6

Nel prototipo Nr 6 abbiamo modificato fortemente la parte mobile contenente telecamere e motori. La struttura a U è stata sostituita da un sostegno di forma rettangolare visto frontalmente, al cui interno sono stati montati i motori asimmetricamente rispetto all'asse di rotazione. In particolare il secondo motore, sottostante la telecamera destra, è stato *appeso* in modo da distribuire il peso uniformemente per ogni angolo di rotazione. Questa scelta si è rivelata molto efficace dato che l'intera struttura ha iniziato a muoversi. Il movimento tuttavia non era ancora sufficiente nè per quanto riguarda l'ampiezza raggiungibile nè per quanto riguarda le velocità raggiungibili, per questo motivo abbiamo provveduto a sostituire il motore esterno con un motore notevolmente più potente¹. Ricordiamo inoltre che per sicurezza abbiamo montato il riduttore fin da subito anche su questo nuovo motore. La struttura così creata aveva un peso di circa 1.5 chilogrammi a cui sono da sottrarre circa 80 grammi per un supporto temporaneo in legno per utilizzare il sistema lontano dal robot Maximum One.

¹circa 7 volte

5.2.4 Prototipo Nr.7

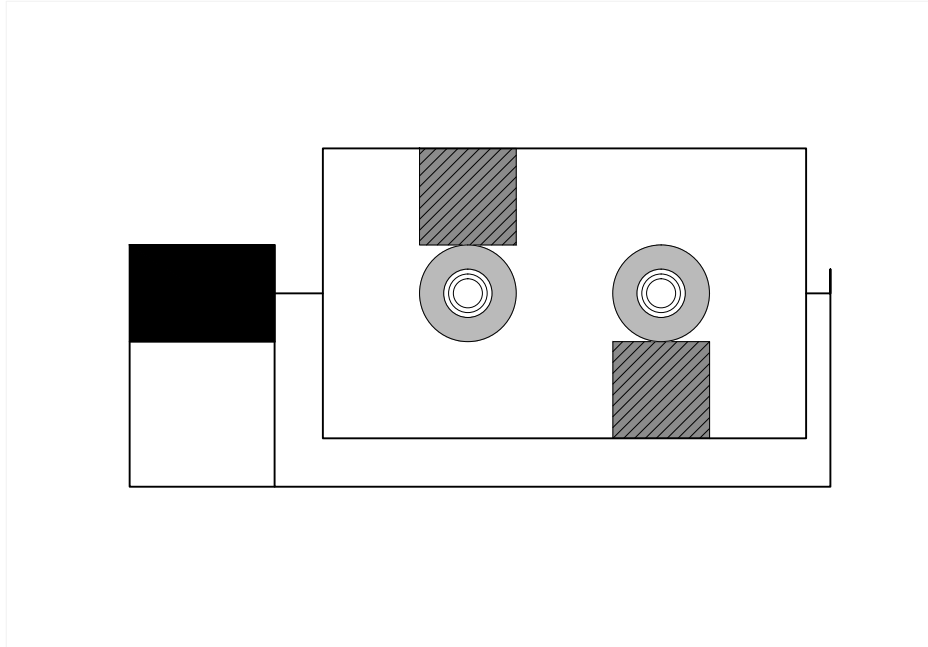


Figura 5.8: Rappresentazione frontale semplificata del prototipo nr.7

Il prototipo Nr.6, seppur funzionante, è stato infine perfezionato nel prototipo Nr.7 rifacendo la base in plexiglas di spessore 0.5 cm, dove erano montate le placchette di sostegno per motore e parte mobile. Tale base è stata ridotta notevolmente al fine di ridurre il peso totale, inoltre l'uso di un materiale parzialmente elastico permette di assorbire sia le vibrazioni inevitabilmente causate dal movimento del motore passo passo per il movimento verticale, sia per assorbire eventuali piccoli disallineamenti dei due assi di rotazione della parte mobile. Al fine di ridurre ulteriormente il peso, la leva della struttura mobile è stata ulteriormente ridotta e la placchetta di sostegno tra motore e parte mobile è stata rimossa. Dopo i primi test preliminari abbiamo eliminato anche il riduttore dato che non comportava un guadagno nè per quanto riguarda lo spostamento del carico (il nuovo motore era sufficiente per questo compito) nè per quanto riguarda la velocità massima (la riduzione infatti dimezzava di fatto la velocità). Abbiamo inoltre sostituito la colla a caldo che fissava le due webcam agli alberi dei motori con del silicone specifico per alte temperature dopo aver assodato²

²con una termocoppia

che mantenendo il sistema acceso per diverse ore il calore sprigionato dai motori raggiungeva i 55° , temperatura sufficiente ad un parziale scioglimento del collante usato per il fissaggio. Un effetto secondario dell'uso del silicone è stato quello di attutire, in misura molto maggiore rispetto alla colla a caldo, le vibrazioni delle telecamere, questo perché una volta asciutto il silicone rimaneva notevolmente elastico al contrario della colla a caldo che una volta indurita rimaneva molto più rigida.

Abbiamo infine montato i 3 sensori di finecorsa; ogni sensore presentava

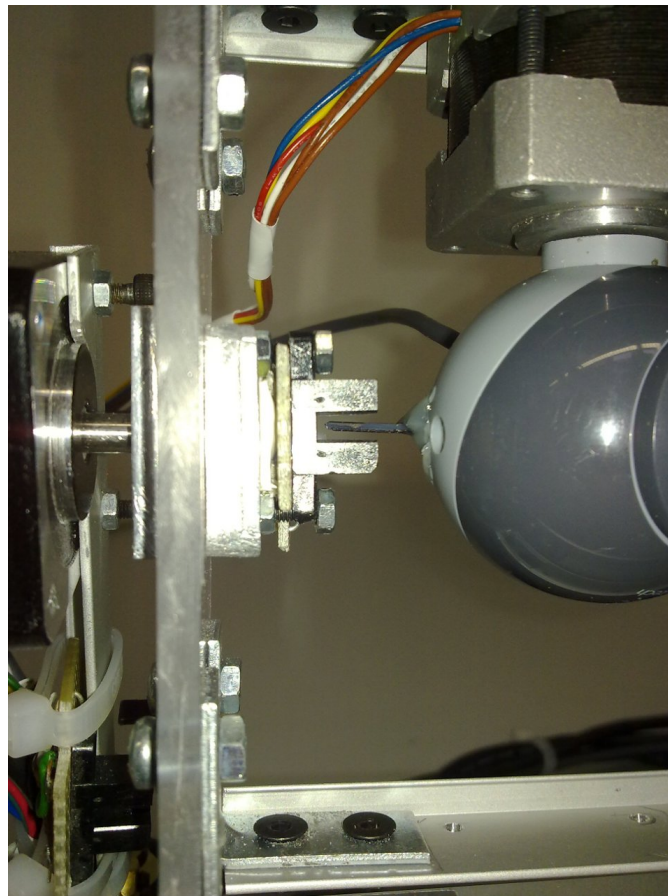


Figura 5.9: Primo piano del sensore di finecorsa montato vicino alla telecamera destra.

una fessura che, in caso fosse ostruita da un oggetto di spessore adeguato, inviava il segnale di raggiungimento del massimo angolo permesso al sistema elettronico. Due sensori sono stati montati lateralmente rispetto alle telecamere ed abbiamo dotato le telecamere di una placchetta in plastica in grado di ostruire completamente lo spazio apposito del sensore come mostrato nelle

figura 5.9. Per quanto riguarda il movimento verticale abbiamo montato un sensore nella parte posteriore, tale sensore veniva ostruito da una placca in plastica fissata alla struttura esterna della parte mobile.

L'immagine della struttura finale è mostrato in figura 5.10. Il peso totale

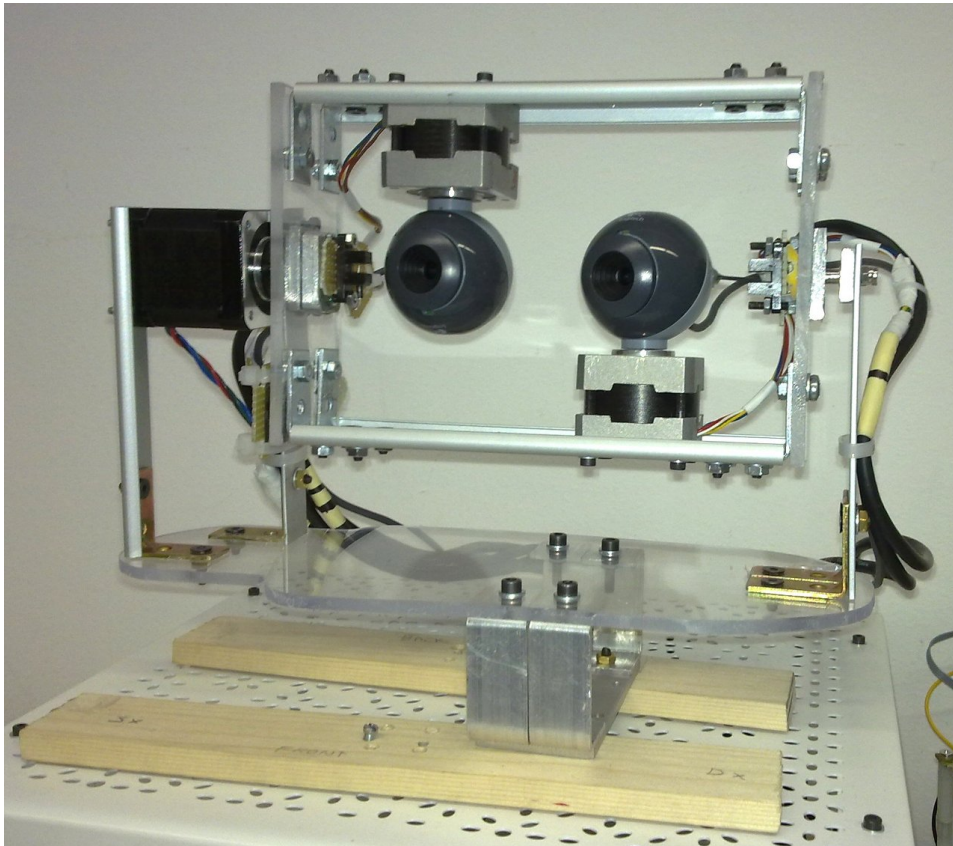


Figura 5.10: Fotografia della versione definitiva della struttura meccanica di MOE.

della struttura è risultato pari a 1.7 kg chiaramente fuori specifiche, a questo dato vanno tuttavia sottratti :

- 80 grammi della base in legno;
- circa 100 grammi di cavi che in caso fossero fissati a corpo del robot Maximum One non scaricherebbero il loro peso direttamente sul collo;
- circa 200 grammi per l'attacco in alluminio di MOE al collo che può essere rimosso e sostituito con dei semplici fori sulla base in plexiglas.

Il peso finale della struttura è quindi di 1.3 chilogrammi; ricordiamo però che circa 200-250 grammi potrebbero essere recuperati nel caso si decidesse

di sostituire il motore verticale con uno più costoso ma dotato di prestazioni maggiori nonché di peso ridotto.

Capitolo 6

Architettura dell'apparato elettronico MOE

“Un giorno le macchine riusciranno a risolvere tutti i problemi, ma mai nessuna di esse potrà porne uno.”

Albert Einstein (1879-1955), fisico tedesco ...

In questo capitolo presenteremo l'apparato elettronico MOE. Esso nasce con lo scopo di permettere un controllo di precisione in posizione e in velocità delle webcam presenti nell'apparato meccanico MOE.

6.1 Introduzione

L'intero progetto si basa sulla scelta della tecnologia dei motori; essa infatti influenza sia l'elettronica di controllo che lo stadio di potenza.

Sebbene l'utilizzo di servomotori potesse sembrare una scelta scontata, in questa sede abbiamo scelto dei motori passo-passo per soddisfare il requisito di precisione presente nelle specifiche. In un'ottica di contenimento costi non sarebbe stato possibile infatti conciliare tale specifica di precisione utilizzando dei servomotori economici. Sono stati determinanti inoltre nella scelta della tecnologia la possibilità di eseguire un controllo ad anello aperto, eliminando in questo modo eventuali problematiche di stabilità, nonché la buona disponibilità di integrati per il controllo.

Abbiamo affidato la logica del controllo a un PIC Microchip della famiglia 16F, che assolve egregiamente la funzione di pilotaggio dei 3 motori stepper, grazie ad una elevata potenza di calcolo e un buon numero di porte I/O. Al PIC vengono affiancati gli integrati L297 e L298. Inoltre la presenza di un

modulo USART nel PIC utilizzato ci ha indotto ad utilizzare una seriale RS232 come interfaccia verso l'esterno. Tale tecnologia è molto utilizzata negli interfacciamenti punto-punto di applicazioni automatiche industriali.

L'apparato elettronico MOE è composto da 4 schede millefori: nella scheda superiore risiede il processore, l'interfaccia RS232 e lo stabilizzatore di alimentazione per TTL LM7805. Nel firmware del processore abbiamo implementato la logica di controllo di più alto livello. Le altre millefori contengono ciascuna la coppia di integrati L297 e L298 e un ponte di diodi ad alta velocità; essi fungono da controllo logico dello stato (controllo delle commutazioni delle fasi) e da controllo di potenza (alimentano le fasi in PWM regolando in anello chiuso tramite un feedback di tensione - circuito chopping). La scheda che alimenta il motore verticale, che necessita di una corrente più elevata rispetto agli altri, differisce dalle altre per via di un dissipatore più grande e di una ventola di raffreddamento, ovvero di un sistema di dissipazione di calore più efficiente.

6.2 Componenti integrati utilizzati

6.2.1 PIC 16F877

Il microprocessore utilizzato è il PIC 16F877-20/P che presenta le caratteristiche di base riassunte nella tabella 6.1:

velocità operativa:	20 Mhz di clock e 200 ns istruzione di ciclo
memoria programma FLASH	8 Kword (8K x 14bit)
memoria dati EEPROM	256 bit
memoria dati RAM	368 bit
n. pins Input / Output	33
n. ADC (risoluzione 10 bit)	8
n. Comparatori	2
n. Timer (risoluzione 8 bit)	2
n. Timer (risoluzione 16 bit)	1
dispositivi integrati utilizzati	AUSART
tensione di alimentazione	[2,5.5] Vdc

Tabella 6.1: Caratteristiche di base del PIC 16F877-20/P

6.2.2 MAX232

Il MAX232 è composto da una coppia di driver/ricevitori e un generatore di tensione capacitivo per erogare una tensione compatibile EIA-232 da un'alimentazione di 5V.

- Ogni ricevitore converte un ingresso EIA-232 in TTL/CMOS da 5V;
- Ogni driver converte un segnale TTL/CMOS in EIA-232.

Un segnale EIA-232 può assumere 3 caratteristiche elettriche:

- 1 logico: se la tensione è compresa tra -3V e -15V;
- 0 logico: compresa tra +3V e +15V;
- Incertezza: compresa tra -3V e +3V.

Un segnale TTL invece assume le seguenti caratteristiche:

- 1 logico: per tensione compresa tra 2V e 5V;
- 0 logico: compresa tra 0V e 0,8V;
- Incertezza: compresa tra 0,8V e 2V.

Per convertire un segnale da EIA-232 a TTL occorre invertirne la tensione e traslarlo al livello di 5V. Similmente per convertire un segnale da TTL a EIA-232 occorre invertirne la tensione e traslarlo secondo il protocollo EIA-232.

L'integrato MAX232 è alimentato a 5V, genera lo 0 logico EIA-232 a 8,5V e l'1 logico a -8,5V. Sebbene disponga di due unità driver/ricevitore, nell'apparato presentato ne viene utilizzato solo una.

6.2.3 LM7805

L'integrato LM7805 è un regolatore di tensione in uscita a 5V 1A massimo. Dispone di un limitatore di corrente in caso di corto circuito e un sistema di protezione termica. La tensione massima in ingresso deve essere inferiore a 35V. Nel progetto del sistema MOE viene utilizzato in configurazione ad uscita fissa.

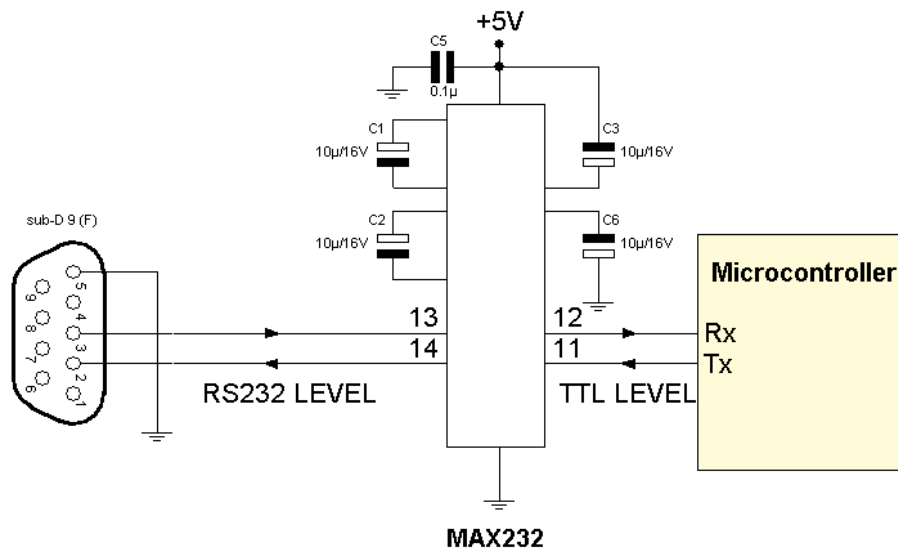


Figura 6.1: Interfacciamento RS232 tramite MAX 232

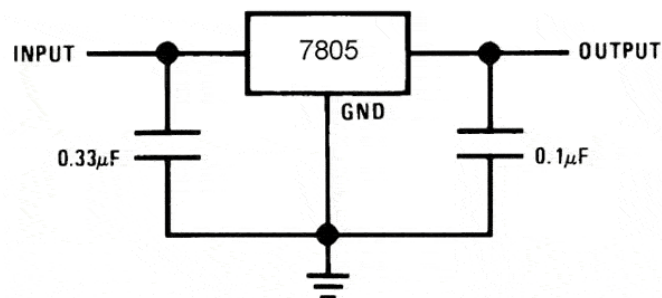


Figura 6.2: Schema elettrico LM7805

6.2.4 Motori passo-passo

I motori passo-passo si dividono in due categorie: a magnete permanente e a riluttanza variabile. I motori a magnete permanente a loro volta si dividono in unipolari o bipolari. Nel progetto sono stati utilizzati 3 motori a magnete permanente bipolari. Essenzialmente essi consistono in un magnete permanente rotante (rotore) e uno statore ai cui poli si trovano avvolgimenti magnetici.

Per il pilotaggio di tali motori viene usata una corrente bipolare, in questo modo è possibile alimentare gli avvolgimenti in sequenza. Il campo

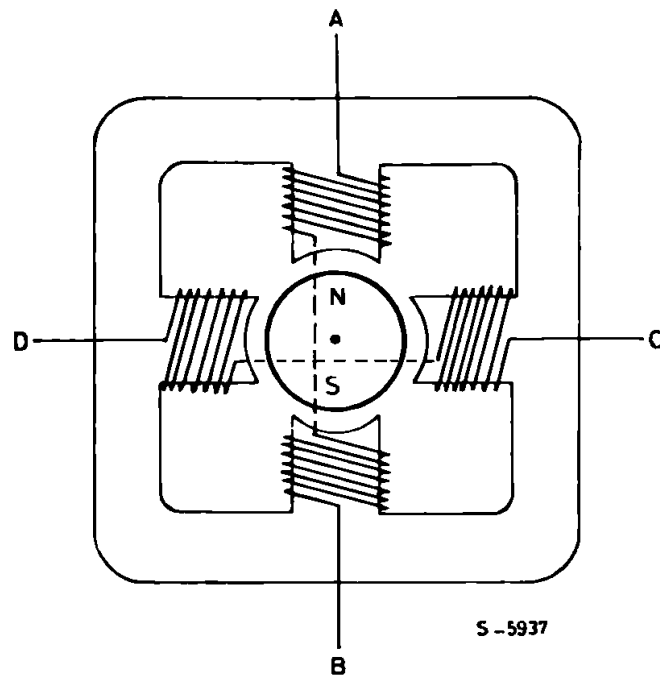


Figura 6.3: Motore passo-passo bipolare

magnetico generato permette al rotore di muoversi allineandosi ai poli dello statore. Per questo tipo di motore ci sono tre possibili sequenze di pilotaggio. La prima possibilità consiste nell'alimentare gli avvolgimenti nella sequenza AB/CD/BA/DC (dove BA e DC significa che gli avvolgimenti AB e CD sono alimentati con polarità inversa). Questa sequenza si chiama *one phase on* o ad onda, e consiste nell'eccitare un avvolgimento alla volta.

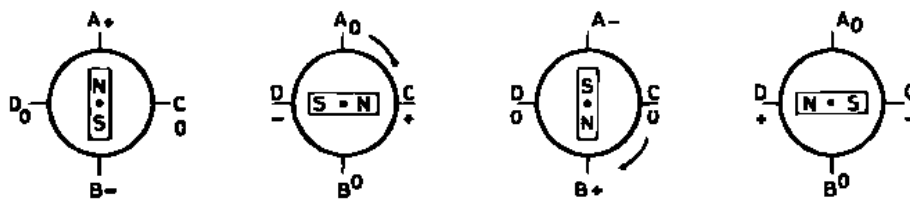


Figura 6.4: Pilotaggio one phase on

Un seconda possibilità è la cosiddetta *two phase on*, nella quale vengono alimentati due avvolgimenti contemporaneamente, così che il rotore si posizioni tra i due poli. Questa modalità costituisce lo standard nel pilotaggio

del motore in quanto si ha la massima coppia.

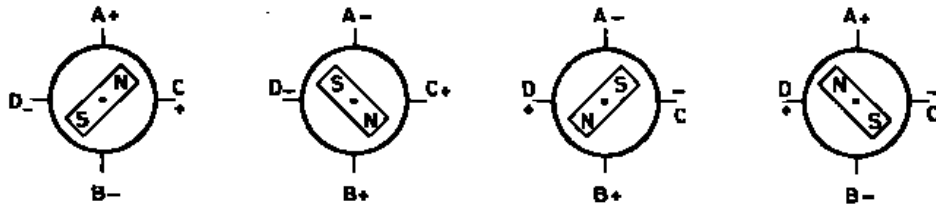


Figura 6.5: Pilotaggio two phase on

La terza possibilità è quella di alimentare in sequenza una fase, poi due, poi una e così via. In questo modo il motore incrementa la posizione di mezzo passo per volta invece che per passi interi. Questa modalità prende il nome di *half step* ed è caratterizzata da un angolo di passo inferiore, oltre che da una coppia minore. Inoltre diminuendo l'angolo di passo occorrono ovviamente più passi rispetto alle sequenze di controllo precedenti (*full step*) per ottenere un dato angolo di spostamento.

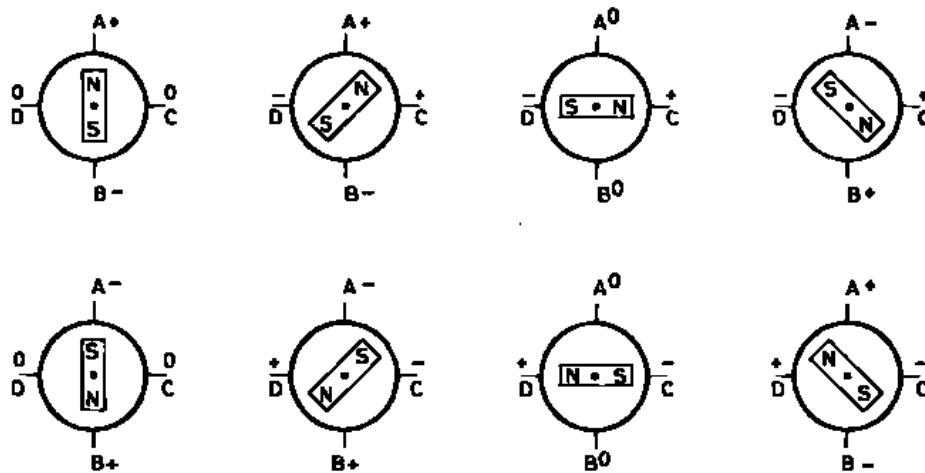


Figura 6.6: Pilotaggio half step

I motori che abbiamo utilizzato nel progetto sono di due tipi:

RS535-0489 (singola unità)

Angolo di passo	1,8°
Precisione sull'angolo di passo	5% (in full step, senza carico)
Tensione di alimentazione:	2,8V
Corrente di alimentazione:	1,68A
Coppia di stallo:	44 Ncm

Tabella 6.2: Specifiche motore RS535-0489

RS440-420 (due unità)

Angolo di passo	1,8°
Precisione sull'angolo di passo	5% (in full step, senza carico)
Tensione di alimentazione:	5V
Corrente di alimentazione:	0,5A
Coppia di stallo:	7 Ncm

Tabella 6.3: Specifiche motori RS440-420

La coppia di motori RS440-420 non sono propriamente dei passo-passo bipolari, ma unipolari ibridi a 6 fili. Essi possono essere comunque utilizzati come bipolari; in questo caso il cablaggio elettrico da effettuare è descritto nell'articolo [24].

6.2.5 L297

L297 integra la circuiteria di controllo per motori bipolari e unipolari. Usato insieme all'L298 forma un'interfaccia completa tra il microcontrollore e un motore bipolare. L297 riceve i segnali dal controllore (nel progetto costituito dal microprocessore PIC) e fornisce i necessari segnali di controllo per lo stadio di potenza (L298); in più fornisce due circuiti per il controllo della corrente in PWM nei due avvolgimenti del motore. E' possibile controllare il motore passo passo sia in *one phase on*, in *two phase on* e anche in *half-step*.

L'uso della coppia di integrati L297 e L298 comporta molti vantaggi. Il software di controllo da implementare risulta semplificato nelle sue operazioni di basso livello (es. gestione delle sequenze di pilotaggio, verso di rotazione, modalità di pilotaggio). Inoltre la possibilità di controllare la corrente di uscita in PWM offre sia alte prestazioni che un'elevata flessibilità in fase di design, dal momento che i vari tipi di motori vengono supportati indipendentemente dalle tensioni e correnti nominali necessarie.

Il controllo di corrente mediante PWM è un controllo ad anello chiuso di tipo on-off. E' realizzato da un comparatore, un flip flop e un resistore sensore in serie all'avvolgimento. Il flip flop viene settato da un oscillatore e quindi produce in uscita una serie di impulsi che comandano lo stadio di potenza. Come conseguenza si ottiene un aumento di corrente negli avvolgimenti. Allo stesso tempo anche sul resistore sensore la tensione risultante aumenta secondo la legge di Ohm. Quando essa supera il valore di riferimento, il comparatore commuta lo stato attivando il reset nel flip flop che quindi smetterà di produrre impulsi.

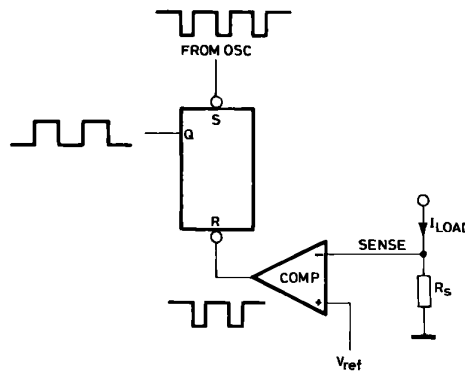


Figura 6.7: Circuito chopper per pilotaggio PWM

Mostriamo l'integrato nella figura 6.8 mentre riportiamo la descrizione dei pin nella tabella 6.4 a pagina 50.

I pin di output A,B,C,D determinano se l'avvolgimento corrispondente viene alimentato e la sua polarità. A e B si riferiscono all'avvolgimento AB.

- Se $A = 0$ e $B = 0$ l'avvolgimento AB non è alimentato.
- Se $A = 1$ e $B = 0$ l'avvolgimento AB è alimentato.
- Se $A = 0$ e $B = 1$ l'avvolgimento AB è alimentato con polarità inversa.

Lo stesso accade per C e D in riferimento all'avvolgimento CD.

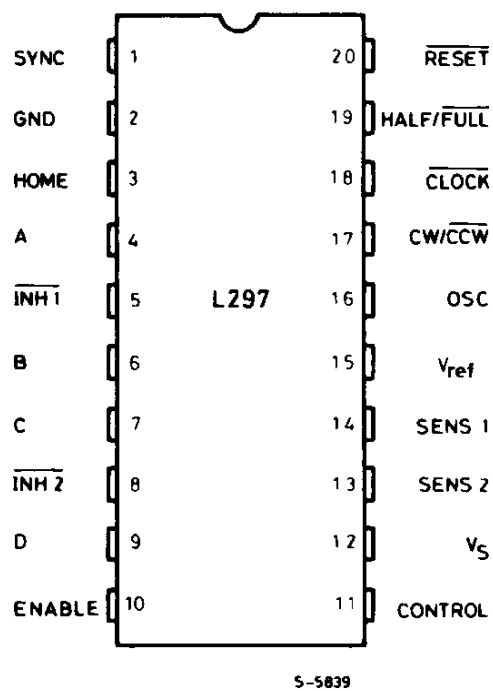


Figura 6.8: Diagramma pin L297

N°PI	NOME	FUNZIONE
1	SYNC	Uscita dell'oscillatore integrato. Input se necessario sincronizzare più L297
2	GND	Massa
3	HOME	Uscita alta quando il sistema si trova nello stato HOME (0101)
4	A	Segnale di pilotaggio fase A per il lo stadio di potenza
5	INH1	Se basso inibisce I transistor sulla fase A e B. L'avvolgimento viene portato a massa per un rapido scarico di corrente
6	B	Segnale di pilotaggio fase B per il lo stadio di potenza
7	C	Segnale di pilotaggio fase C per il lo stadio di potenza
8	INH2	Se basso inibisce I transistor sulla fase C e D. L'avvolgimento viene portato a massa per un rapido scarico di corrente
9	D	Segnale di pilotaggio fase D per il lo stadio di potenza
10	ENABLE	Se basso A,B,C,D,INH1 e INH2 vengono settati bassi
11	CONTROL	Se alto il circuito chopper agisce in A,B,C,D; se basso il chopping viene ottenuto utilizzando INH1, INH2. Vedi paragrafo relativo a L298
12	Vs	Alimentazione 5V
13	Sens2	Tensione sul resistore sensore dell'avvolgimento CD
14	Sens1	Tensione sul resistore sensore dell'avvolgimento AB
15	Vf	Riferimento di tensione sul resistore sensore. Determina la corrente di picco sugli avvolgimenti
16	OSC	Costante di tempo per l'oscillatore integrato
17	CW/CCW	Input che setta il verso di rotazione del rotore. Esso dipende da come si collegano gli avvolgimenti
18	CLOCK	Ad ogni fronte di salita corrisponde uno step
19	HALF/FULL	Input che determina il pilotaggio half-step o full-step. Se basso è in full.
20	RESET	A fronte di un impulso basso porta il sistema nello stato HOME

Tabella 6.4: Tabella sulla descrizione dei pin dell'integrato

Segue il diagramma degli stati per il pilotaggio di un motore bipolare.

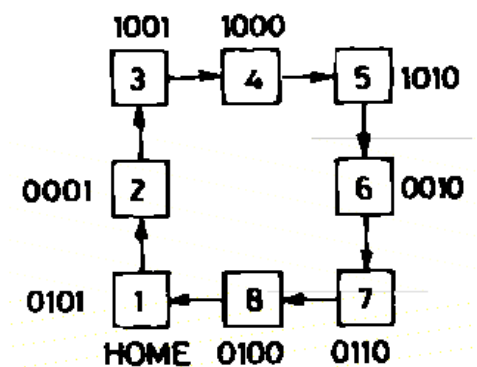


Figura 6.9: Diagramma stati pilotaggio di un passo-passo

Il pilotaggio *two phase on* si ottiene con la seguente sequenza:

1 ; 3 ; 5 ; 7 ; 1

Il pilotaggio *half-step* si ottiene con la seguente sequenza:

1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 1

Il pilotaggio *one phase on* si ottiene con la seguente sequenza:

2 ; 4 ; 6 ; 8 ; 2

La logica con cui L297 gestisce gli stati degli avvolgimenti è abbastanza semplice:

```

1 se HALF/FULL = 0 e CW/CCW = 0,
2     prossimo_stato = stato_attuale + 2
3 se HALF/FULL = 0 e CW/CCW = 1,
4     prossimo_stato = stato_attuale - 2
5 se HALF/FULL = 1 e CW/CCW = 0,
6     prossimo_stato = stato_attuale + 1
7 se HALF/FULL = 0 e CW/CCW = 0,
8     prossimo_stato = stato_attuale - 1
  
```

Essendo il sistema all'accensione nello stato HOME, si evince che per pilotare il motore in *one phase on* è necessario spostarsi in uno degli stati pari con un mezzo step, altrimenti se si pilota in full-step lo si farà in *two phase on* (chiamata anche normal driving - in quanto è la modalità di pilotaggio full step predefinita per motori bipolari).

Ulteriori informazioni sono reperibili nell'articolo [27].

6.2.6 L298 e pilotaggio in PWM

L'integrato L298 è un doppio ponte H con un controllo logico sui transistor per migliorare prestazioni e affidabilità. Può essere alimentato con tensione fino a 46V e dispone di una seconda alimentazione per i circuiti logici. Esso eroga una corrente fino a 4A (2A per ogni ponte) e dispone di una protezione per surriscaldamento.

Un ponte H è costituito da 4 interruttori secondo il seguente schema.

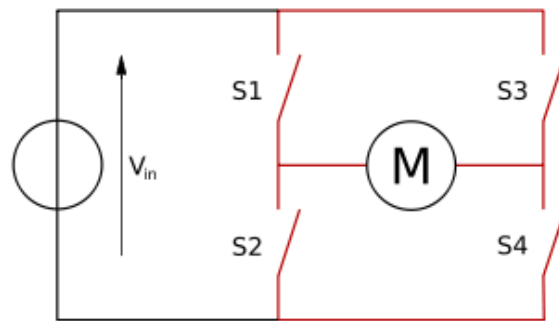


Figura 6.10: Circuito ideale ponte H

Quando gli interruttori S1 e S4 sono chiusi (e S2 e S3 aperti) una tensione positiva viene applicata sul carico M; mentre se sono aperti S1 e S4 e chiusi S2 e S3 la tensione è invertita. S1 e S2 non possono essere aperti nello stesso tempo, per evitare di corto circuitare l'alimentatore; analogamente S3 e S4.

Nella figura 6.11 viene mostrato lo schema elettrico di uno dei due ponti H integrati nel L298. Gli interruttori sono 4 transistor mentre il carico è costituito dall'avvolgimento del motore passo-passo. I diodi (non integrati nel componente) D1,D2,D3 e D4 sono ad alta velocità (Shottky) e servono per proteggere i transistor dalle correnti di *fly-back* generate dall'avvolgimento non appena si interrompe l'alimentazione. Le 4 porte logiche AND pilotano le basi dei transistor e non permettono di corto circuitare l'alimentatore.

Gli ingressi A e B (e INH1) sono pilotati da un L297 e può verificarsi uno dei seguenti casi:

A = 0 e B = 1 (e INH1 =1) in questo caso conducono i transistor Q2 e Q3 e viene applicata al carico una tensione positiva;

A = 1 e B = 0 (e INH1 =1) in questo caso conducono i transistor Q1 e Q4 e viene applicata al carico una tensione negativa.

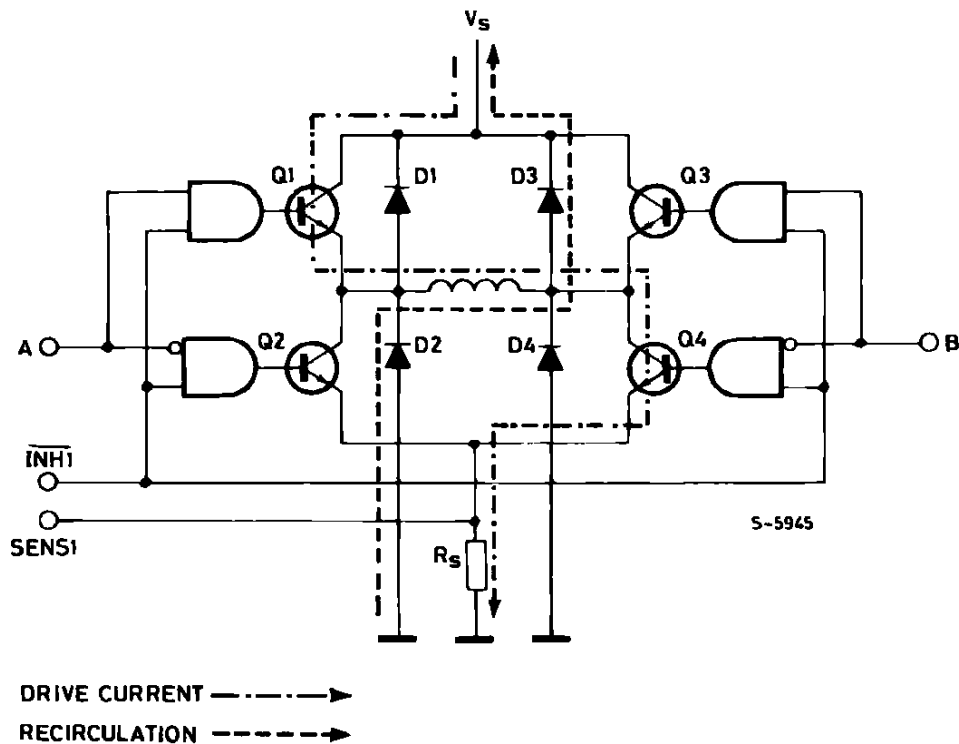


Figura 6.11: Ponte H, nel L298

L'ingresso INH1 di norma è impostato alto, se è basso interdice la conduzione dei quattro transistor. Il motivo di questo comportamento è scaricare la corrente dell'avvolgimento nel resistore R_s , in questo modo si ottiene un tempo di scarica minore, stressando il resistore. In condizioni normali, quando l'avvolgimento non è più alimentato le correnti di ricircolo attraversano i diodi, per scaricarsi sul polo positivo, il tempo di scarica è maggiore.

L298 è adatto ad essere pilotato in PWM. Questa soluzione costituisce lo stato dell'arte nel pilotaggio di motori passo passo bipolari.

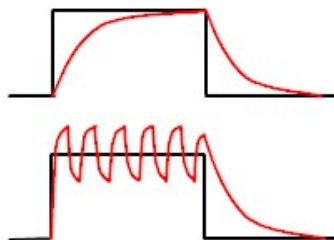


Figura 6.12: Pilotaggio tradizionale e pilotaggio PWM

Come possiamo intuire dalla figura 6.12 il pilotaggio PWM consiste nell'utilizzare un'alimentazione molto più elevata di quella richiesta dal motore; quando la corrente raggiunge il valore nominale, il transistor viene aperto e quindi la corrente comincia a diminuire, passando attraverso il diodo di ricircolo. Dopo un piccolo tempo il transistor viene di nuovo chiuso, la corrente comincia di nuovo ad aumentare ancora fino al raggiungimento della corrente nominale, e così via. Se l'alternarsi delle fasi di apertura/chiusura del transistor sono molto più veloci della velocità con cui le fasi sono eccitate, il valore medio della corrente è praticamente uguale alla corrente nominale.

Il pilotaggio PWM ha due vantaggi: il tempo di salita della corrente molto elevato (visto che è elevata la tensione di alimentazione) e, durante le fasi di spegnimento del transistor non si ha consumo di corrente.

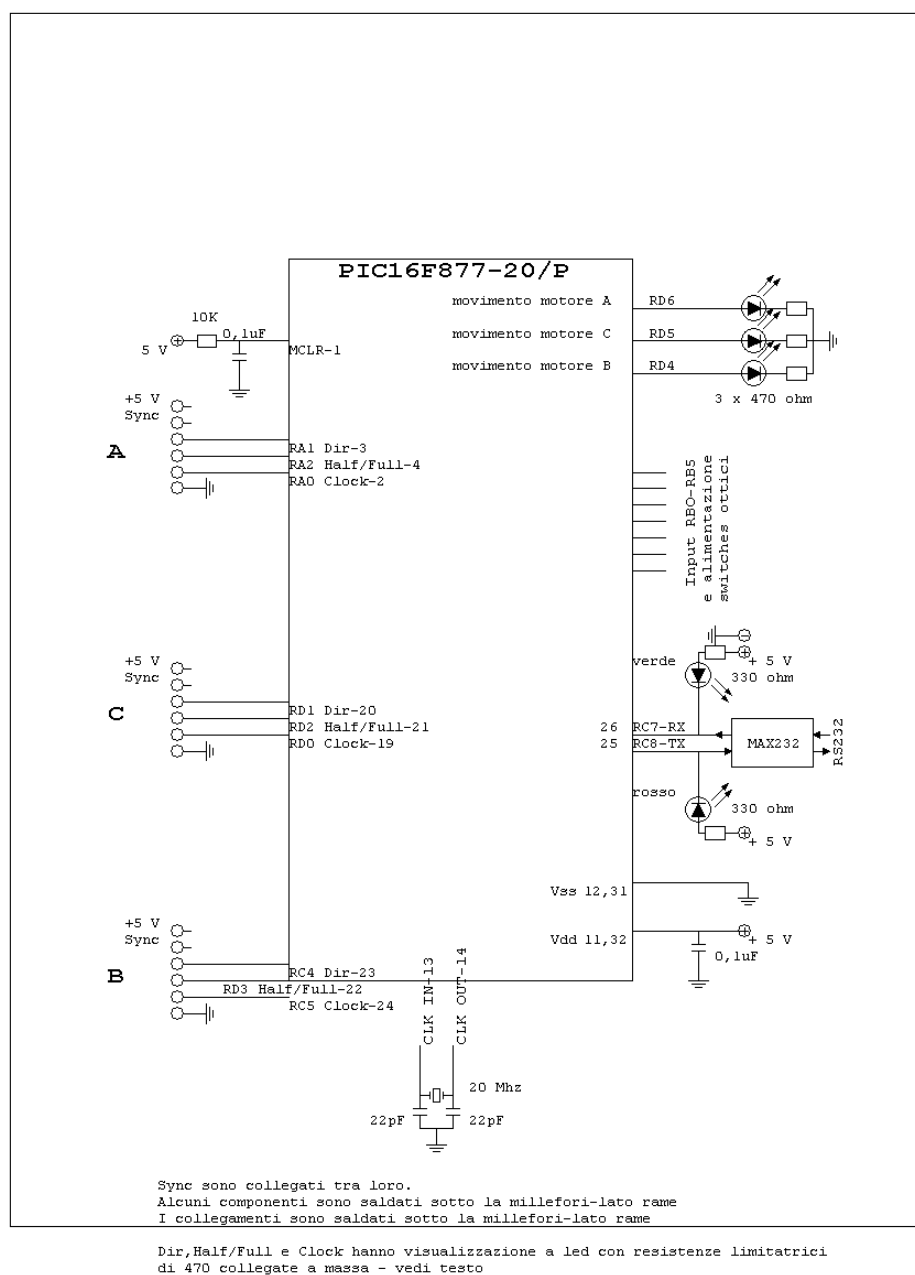
Ulteriori informazioni sono reperibili nell'articolo [28]

6.3 Realizzazione apparato elettronico MOE

In questa sezione illustreremo gli schemi elettrici implementati, che costituiscono l'attuale elettronica dell'apparato. Abbiamo diviso le 4 schede implementate in due categorie: scheda microprocessore e schede di pilotaggio. Presentiamo le tre schede di pilotaggio tramite un'unico schema elettrico, sebbene nel progetto esse non siano esattamente equivalenti. Tuttavia le tre schede di pilotaggio differiscono solo per caratteristiche marginali quali l'alimentazione dei ponti H e la grandezza e tipologia del dissipatore del circuito di potenza.

6.3.1 Schema elettrico scheda microprocessore

Il microchip e tutti i circuiti TTL vengono alimentati tramite un alimentatore switching. La tensione di uscita dallo switching è 12V, portati a 5 grazie allo stabilizzatore LM7805. Tale tensione alimenta il MAX232(interfaccia RS232- TTL), il microcontrollore PIC16F877-20/P, e le tre schede controllori dei motori stepper che montano i tre integrati L297 ed i tre integrati L298. Il cablaggio del 7805 è fatto su circuito stampato, dove trova alloggio anche l'integrato MAX232 che ha la funzione di interfacciare l'RS232 al microprocessore che ha ingressi ed uscite a livello TTL. Il circuito stampato



che supporta lo stabilizzatore 7805 ed il MAX232 è a sua volta fissato su una basetta millefori in cui è assemblato il PIC16F877-20/P che ha i seguenti collegamenti:

- A,C,B da 6 pin ognuno per le tre schede millefori che controllano i motori stepper bipolari;
- $6 * 4$ pin per i sensori di fine e inizio corsa;
- 2 pin in parallelo + 12 Volt per alimentare i tre ponti H dei tre integrati L298;
- una presa tipo coassiale da collegare all'alimentatore switching 12 Volt.

Ogni presa di sei poli (A - B - C) collegate alle schede controllori motori stepper sono così configurate:

- +5 Volt
- Sync (sincronizzazione degli oscillatori dei tre chopper negli L297);
- Direct (segnalato dal led giallo) se low = ccw;
- Half/Full (segnalato dal led verde) se low = full step ($1,8^\circ$);
- Clock (segnalato dal led rosso) ogni fronte di salita è uno step o mezzo step a seconda del livello di Half/Full;
- Gnd.

Se le prese descritte vengono inserite al contrario gli integrati alimentati nelle schede di pilotaggio andrebbero in corto circuito. Per evitare questo abbiamo inserito un diodo di protezione su ciascuna delle tre schede e messo un fusibile da 500 mA all'uscita dell'integrato LM7805.

E' stato necessario implementare un sistema di taratura dei fine corsa dei motori, da utilizzare sia nel caso di startup del sistema, sia nel caso di un'eventuale perdita di passo durante uno spostamento. Per tale scopo abbiamo utilizzato degli switch ottici TCST 2103 della Telefunken con led emittente a 950 nm, Vcc 5V e IF 50 mA. Alimentando il led a 5 Volt, la resistenza limitatrice in serie al Led sarà: $R = \frac{V_{cc} - V_F}{I_F}$, dove V_F è la Forward Voltage che è 1,2 Volt, per cui R è di circa 220 Ohm. Il dimensionamento della resistenza in serie al collettore non è critico considerato l'ingresso ad alta impedenza del Pic e può essere ad esempio di 10 KOhm. Lo schema del sensore ottico implementato è il seguente:

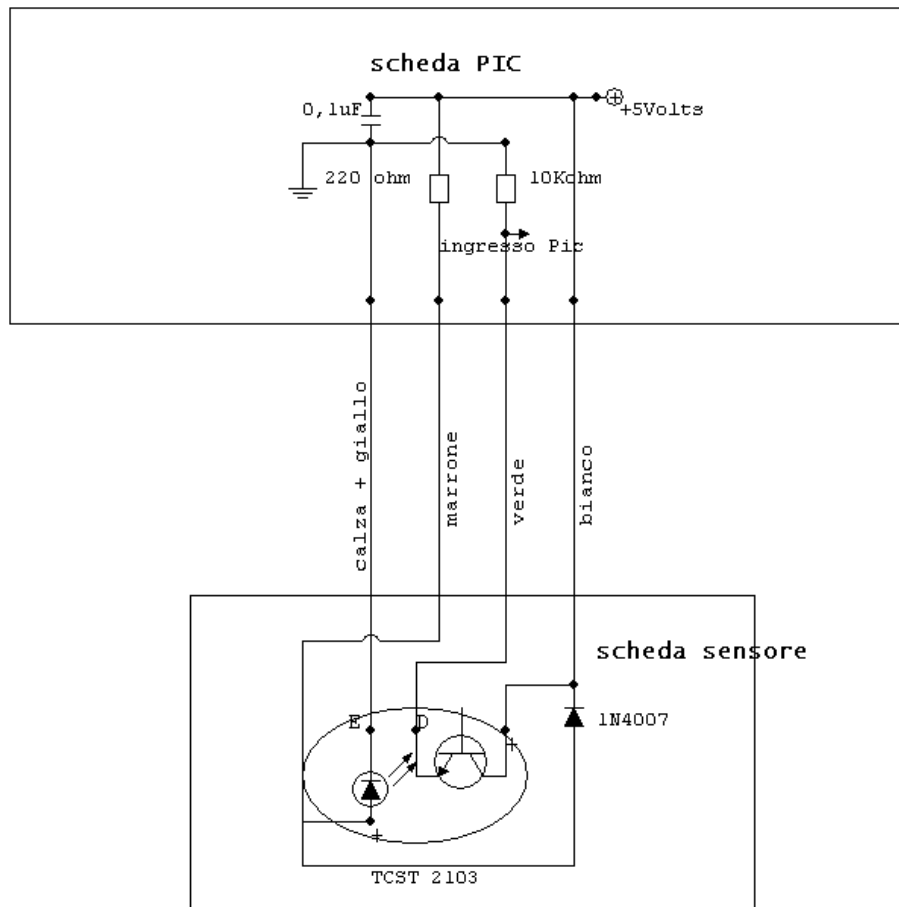


Figura 6.14: Schema elettrico sensore ottico

6.3.2 Schema elettrico della scheda di pilotaggio

Vref viene alimentato con un partitore resistivo tarato per far fornire ai motori RS440-420 una corrente di 0,5 A come da datasheet. Al piedino 4 dell'integrato L298 arriva una tensione di 12 Volt fornita direttamente dall'alimentatore switching. Questa tensione serve ad alimentare i ponti H che forniscono tensione ai motori stepper. Per evitare problemi di rumore di massa dovuto a configurazioni multiple, abbiamo adottato, come consigliato dal datasheet, la sincronizzazione degli oscillatori chopper, per cui abbiamo lasciato che un solo L297 generasse il segnale poi distribuito agli altri due attraverso il pin 1 (Sync) collegato in serie. Inoltre i pin 16 (oscillatore) dei due L297 ricevuti la sincronizzazione sono stati messi a massa.

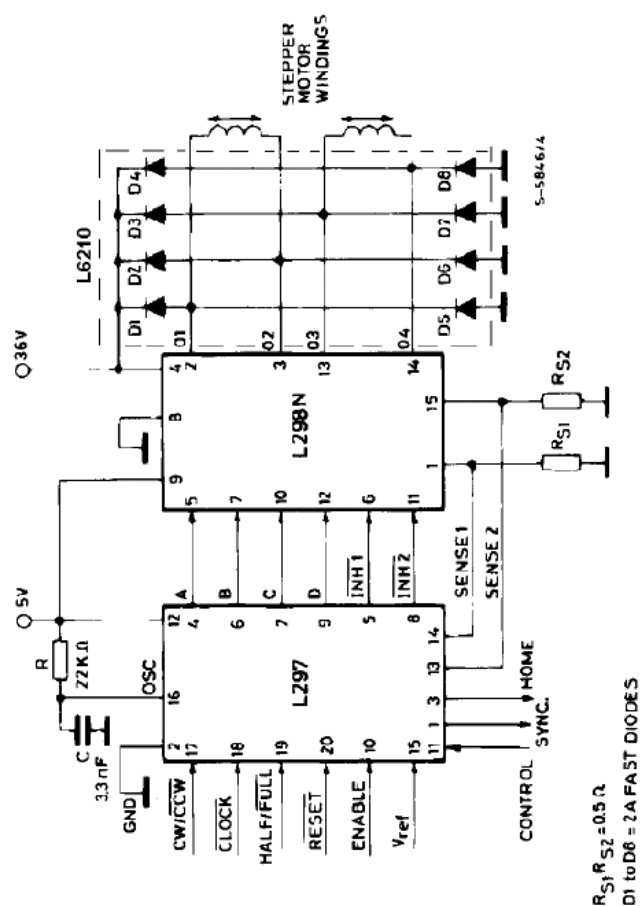


Figura 6.15: Schema elettrico pilotaggio motore passo-passo

Il discorso è leggermente differente per quanto riguarda il pilotaggio del motore RS535-0489. Il partitore di tensione in questo caso regola il circuito chopper ad erogare una corrente di 1,68 A. L'alimentazione dei ponti H non viene erogata dall'alimentatore switching ma da un trasformatore aggiuntivo, mentre la logica TTL viene comunque alimentata dallo switching.

Capitolo 7

Firmware MOE

“La disumanità del computer sta nel fatto che, una volta programmato e messo in funzione, si comporta in maniera perfettamente onesta.”

Isaac Asimov (1920-1992), scrittore americano ...

In questo capitolo presenteremo il firmware MOE. Esso nasce con lo scopo di permettere la gestione della posizione delle telecamere e l'interfacciamento con il mondo esterno.

7.1 Introduzione

Lo scopo del firmware MOE è duplice: implementare l'interfacciamento tra l'apparato elettronico MOE e il mondo esterno e gestire lo stato del sistema meccanico (ovvero la posizione delle webcam). Queste due funzionalità vengono realizzate tramite due task concorrenti. Abbiamo realizzato l'interfacciamento tramite seriale RS232, per via del modulo USART integrato nel processore, proponendo un semplice ma robusto protocollo di comunicazione punto-punto. La routine di gestione dello stato viene presentata come l'implementazione di una macchina a stati.

Il firmware è stato sviluppato su PIC16F877 utilizzando l'IDE freeware MPLAB della Microchip, codificato in C e compilato con la versione lite del compilatore HI-TECH.

Ulteriori informazioni sul PIC utilizzato possono essere trovate nell'articolo [18].

7.2 RS232

L'interfaccia seriale RS232 permette lo scambio di dati tra dispositivi digitali attraverso una comunicazione seriale asincrona. Il termine seriale significa che i bit che costituiscono l'informazione sono trasmessi uno alla volta su di un solo *filo*. Questo termine è in genere contrapposto a *parallelo*: in questo caso i dati sono trasmessi contemporaneamente su più fili, ad esempio 8, 16 o 32. Il termine asincrono significa, in questo contesto, che i dati sono trasmessi senza l'aggiunta di un segnale di clock, cioè di un segnale comune che permette di sincronizzare la trasmissione con la ricezione; ciò nonostante sia il trasmettitore che il ricevitore devono comunque essere dotati di un clock locale per poter interpretare i dati. La sincronizzazione dei due clock è necessaria ed è fatta in corrispondenza della prima transizione sulla linea dei dati. Le unità di misura della velocità di trasmissione sono essenzialmente due: il baud ed il bit per secondo (bps o, meno spesso, b/s).

Il baud rate indica il numero di transizioni al secondo che avvengono sulla linea;

Il bps indica, come dice il nome, quanti bit al secondo sono trasmessi lungo la linea.

Nel caso di trasmissione binaria (dove è presente un livello alto ed uno basso) le due cose ovviamente coincidono numericamente, da cui la parziale equivalenza dei due termini. Ai fini del progetto, essendo la trasmissione esclusivamente binaria, utilizzeremo le due grandezze indifferentemente.

La comunicazione tramite RS232 si instaura tra un trasmettitore e un ricevitore. Essa può essere half duplex o full duplex.

Half Duplex indica che la trasmissione è bidirezionale ma non avviene contemporaneamente nelle due direzioni: un dispositivo (ricevitore, listener o Rx) ascolta e l'altro (trasmettitore, talker o Tx) emette segnali. Quando è necessario si scambiano i ruoli;

Full Duplex indica che la trasmissione è bidirezionale e contemporanea. In questo caso sono necessari ovviamente due fili oppure qualche altro sistema per distinguere i due messaggi contemporanei nelle due direzioni.

Lo standard RS232 permette una trasmissione full-duplex in quanto è utilizzato un conduttore separato per ciascun verso di trasmissione delle informazioni. Il vincolo è in genere la necessità che trasmissione e ricezione abbiano lo

stesso formato e, ovviamente, che ciascuno dei due nodi abbia sufficiente potenza di calcolo per la gestione del duplice flusso di informazioni.

L'ampiezza del segnale è caratterizzata da un valore *alto* pari a circa +12V ed un valore *basso* pari a -12V. L' *uno* logico (spesso indicato come mark) corrisponde al valore di -12V, mentre lo *zero* logico (space) come +12V. Tutte le transizioni appaiono in corrispondenza di multipli di 104 μ s (pari ad 1/9600 cioè ciascun bit dura esattamente l'inverso del baud rate). La linea si trova inizialmente nello stato di riposo, bassa (nessun dato in transito); la prima transizione da basso in alto indica l'inizio della trasmissione (inizia il *bit di start*, lungo esattamente 104 μ s). Segue il bit meno significativo (LSB), dopo altri 104 μ s il secondo bit, e così via, per otto volte, fino al bit più significativo (MSB). Da notare che il byte è trasmesso *al contrario*, cioè va letto da destra verso sinistra. Segue infine un periodo di riposo della linea di almeno 208 μ s, cioè due bit di stop e quindi (eventualmente) inizia un nuovo pacchetto di bit.

Le varianti possibili sono le seguenti:

- Se la trasmissione è più veloce o più lenta, la distanza tra i fronti varia di conseguenza (p.e. a 1200 bps le transizioni avvengono a multipli di 0,833 ms, pari a 1/1200);
- Invece di trasmettere 8 bit, se ne possono trasmettere 6, 7 o anche 9 (ma quest'ultima possibilità non è prevista dalle porte seriali dei normali PC);
- Alla fine è possibile aggiungere un bit di parità;
- Alla fine la linea rimane nello stato di riposo per almeno 1 o 1.5 o 2 bit; è importante notare che, se non ho più nulla da trasmettere, il *riposo* è molto più lungo, ovviamente.

Lo standard originale prevede una velocità fino a 20Kbps.

Oltre ai bit dei dati (in numero variabile tra 5 ed 9) viene inserito un bit di parità (opzionale) per verificare la correttezza del dato ricevuto. Esistono cinque tipi di parità:

None: nessun tipo di parità, cioè nessun bit aggiunto;

Pari (even): il numero di mark (incluso il bit di parità) è sempre pari;

Dispari (odd): il numero di mark (incluso il bit di parità) è sempre dispari;

Mark: il bit di parità vale sempre mark;

Space: il bit di parità vale sempre space.

7.2.1 Modulo USART nel PIC16F877

Il modulo USART¹ è uno dei due moduli di I/O seriale che possiede il PIC16F877. Le modalità di funzionamento della periferica sono le seguenti: asincrona full-duplex (per comunicare con terminali dispositivi periferici quali terminali CRT e PC), sincrona-master half-duplex e sincrona-slave half-duplex (per comunicare con periferiche come circuiti integrati A/D e D/A o EEPROM seriali ecc.).

I registri che controllano la trasmissione, la ricezione e lo stato della periferica USART sono il TXSTA e il RCSTA; i pin utilizzati sono RC6 come trasmissione e RC7 come ricezione. Per il corretto funzionamento bisogna valorizzare il registro SPBRG, che comanda il generatore di baudrate (in pratica un timer e un divisore).

Inserendo un opportuno valore è possibile impostare programmaticamente la velocità di trasmissione. Per il calcolo di tale valore sono state utilizzati i dati forniti dalla Microchip e presenti nella tabella 7.1.

BRGH=1, Fosc=4MHz		BRGH=1, Fosc=8MHz	
Baud	SPBRG	Baud	SPBRG
1200	207.3	1200	415.7
2400	103.2	2400	207.3
4800	51.1	4800	103.2
9600	25.0	9600	51.1
19200	12.0	19200	25.0
38400	5.5	38400	12.0
57600	3.3	57600	7.7
115200	1.2	115200	3.3

BRGH=1, Fosc=16MHz		BRGH=1, Fosc=20MHz	
Baud	SPBRG	Baud	SPBRG
9600	103.2	9600	129.2
19200	51.1	19200	64.1
38400	25.0	38400	31.5
57600	16.3	57600	20.7
115200	7.7	115200	9.8
230400	3.3	230400	4.4
460800	1.7		

Tabella 7.1: Tabella fornita dalla Microcip

¹Universal Synchronous Asynchronous Receiver Transmitter

Pin Diagram

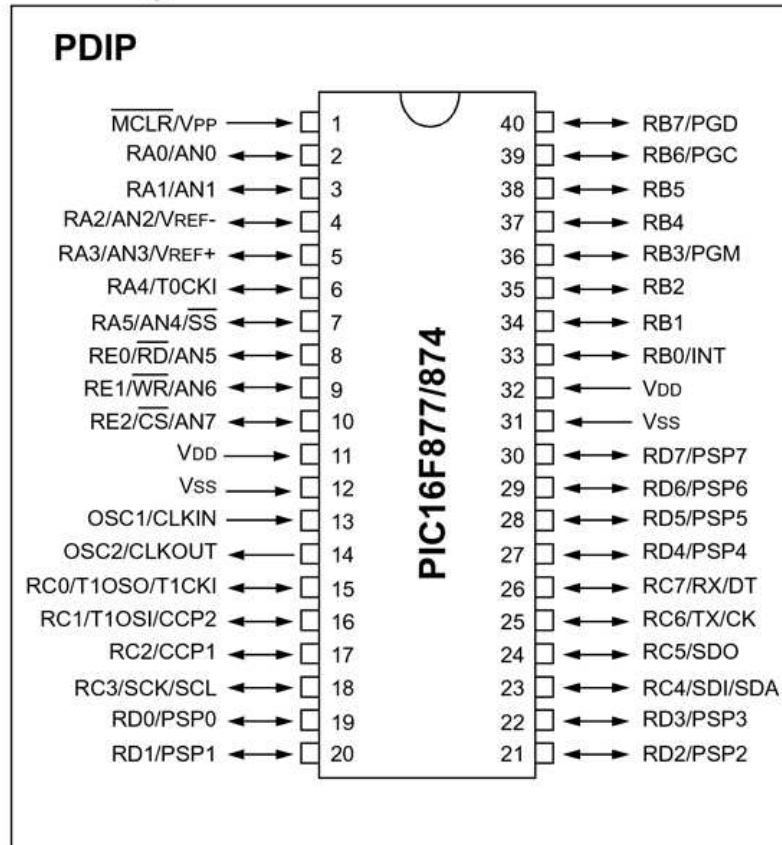


Figura 7.1: Diagramma PIN PIC16F877

Ai fini del progetto abbiamo scelto una comunicazione a 19.200 baud; quindi il registro SPBRG è stato impostato a 64.1. Abbiamo inoltre impostato la comunicazione a 8 bit in trasmissione/ricezione asincrona, senza controllo di flusso. Programmaticamente la trasmissione/ricezione da RS232, grazie al modulo USART nel PIC16F877, si riduce a una lettura/scrittura dei registri TXREG e RCREG.

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

bit 7

bit 0

- bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled
- bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception
- bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care
- bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive
- bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit
- bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error
- bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error
- bit 0 **RX9D:** 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

- n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

Figura 7.2: Registro RCSTA

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

- bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
 Don't care
Synchronous mode:
 1 = Master mode (clock generated internally from BRG)
 0 = Slave mode (clock from external source)
- bit 6 **TX9:** 9-bit Transmit Enable bit
 1 = Selects 9-bit transmission
 0 = Selects 8-bit transmission
- bit 5 **TXEN:** Transmit Enable bit
 1 = Transmit enabled
 0 = Transmit disabled
- Note:** SREN/CREN overrides TXEN in SYNC mode.
- bit 4 **SYNC:** USART Mode Select bit
 1 = Synchronous mode
 0 = Asynchronous mode
- bit 3 **Unimplemented:** Read as '0'
- bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
 1 = High speed
 0 = Low speed
Synchronous mode:
 Unused in this mode
- bit 1 **TRMT:** Transmit Shift Register Status bit
 1 = TSR empty
 0 = TSR full
- bit 0 **TX9D:** 9th bit of Transmit Data, can be parity bit

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figura 7.3: Registro TXSTA

7.3 Protocollo di comunicazione

Il protocollo di comunicazione implementato prevede due differenti pacchetti, il primo che chiamiamo `msg_sent` è definito come segue:

```
1 struct msg_sent {  
2     unsigned char sync0;  
3     unsigned char sync1;  
4     unsigned char pos_y;  
5     unsigned char pos_x1;  
6     unsigned char pos_x2;  
7     unsigned char checksum;  
8     unsigned char sync2;  
9 };
```

Questo pacchetto è trasmesso dal PIC ad un target esterno, nel nostro caso il PC, ed ha il compito di veicolare le informazioni sulla posizione delle telecamere. La sua dimensione è di 7 Byte; i primi due Byte (`sync0` e `sync1`) devono essere valorizzati con gli esadecimali `0xAF` e `0xFA` rispettivamente e fungono da handshaking tra chi riceve e il pacchetto inviato. I tre Byte `pos_y`, `pos_x1` e `pos_x2` sono il corpo del messaggio e contengono le posizioni delle due telecamere nei rispettivi assi. Il Byte di checksum invece contiene lo XOR logico dei tre Byte `pos_y`, `pos_x1` e `pos_x2`. Esso viene controllato da chi riceve per validare l'integrità del pacchetto ricevuto. Il Byte `sync2` è valorizzato a `0xFF` e delimita la fine del pacchetto. Il secondo pacchetto del protocollo, chiamato `msg_rcvd`, è definito come segue:

```
1 struct msg_rcvd {  
2     unsigned char sync0;  
3     unsigned char sync1;  
4     unsigned char id;  
5     unsigned char pos_y;  
6     unsigned char pos_x1;  
7     unsigned char pos_x2;  
8     unsigned char spd_y;  
9     unsigned char spd_x1;  
10    unsigned char spd_x2;  
11    unsigned char half_y;  
12    unsigned char half_x1;  
13    unsigned char half_x2;  
14    unsigned char checksum;  
15    unsigned char sync2;  
16 };
```

Questo pacchetto viaggia dal PC al PIC ed ha il compito di veicolare le informazioni presenti nella tabella 7.2:

id	comando richiesto
pos_y	riferimento di posizione asse y
pos_x1	riferimento di posizione asse x, webcam 1
pos_x2	riferimento di posizione asse x, webcam 2
spd_y	riferimento di velocità motore asse y
spd_x1	riferimento di velocità asse x, motore 1
spd_x2	riferimento di velocità asse x, motore 2
half_y	flag per half step motore asse y (se 1 half step, altrimenti full step)
half_x1	flag per half step motore 1 asse x (se 1 half step, altrimenti full step)
half_x2	flag per half step motore 2 asse x (se 1 half step, altrimenti full step)

Tabella 7.2: Informazioni contenute nel *msg_rcvd*

I campi restanti sono equivalenti al pacchetto *msg_sent*. La sua dimensione è di 14 Byte. Il corpo del messaggio (con cui ricavare la checksum) è costituito dai Byte descritti sopra.

Il protocollo è asincrono, master-slave, dove il PC assume il ruolo di master e il PIC di slave. La trasmissione è chiaramente punto-punto senza ritrasmissione dei pacchetti non validi: essi vengono scartati se non passano il controllo di checksum oppure se non vengono rispettati i segnali di sincronizzazione. Il funzionamento della routine di parsing dei messaggi viene illustrata tramite un diagramma degli stati.

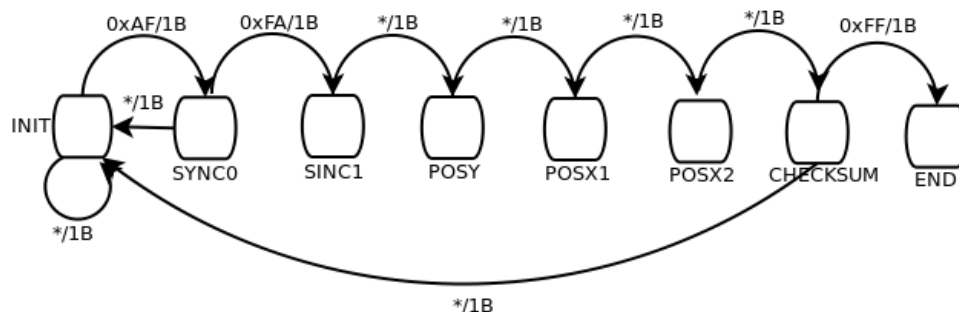


Figura 7.4: Diagramma stati Parser

Si evincono i seguenti stati:

INIT: stato iniziale. Se ricevo il Byte 0xAF passo allo stato SYNC0. Altrimenti rimango nello stato attuale;

SYNC0: Ho ricevuto il primo Byte di handshaking. Se ricevo il Byte 0xFA passo allo stato SYNC1. Altrimenti torno allo stato INIT;

SYNC1: Ho ricevuto il secondo Byte di handshaking. Attendo il Byte POS_Y per passare allo stato POSY;

POSY: Ho ricevuto il Byte POS_Y. Attendo il Byte POS_X1 per passare allo stato POSX1;

POSX1: Ho ricevuto il Byte POS_X1. Attendo il Byte POS_X2 per passare allo stato POSX2;

POSX2: Ho ricevuto il Byte POS_X2. Attendo il Byte di checksum per passare allo stato CHECKSUM;

CHECKSUM: Ho ricevuto il Byte checksum. Attendo il Byte di sync2 0xFF per passare allo stato finale. Altrimenti torno allo stato INIT e scarto le informazioni ricevute;

SYNC2: Ho ricevuto il Byte 0xFF. Sono nello stato finale. Il messaggio è correttamente ricevuto, posso passare al controllo di validità del contenuto dello stesso.

Nel diagramma mostriamo il parsing di un pacchetto di tipo msg_sent, il comportamento del parser del pacchetto msg_rcvd segue lo stesso principio di funzionamento.

7.4 Comandi implementati

Abbiamo sviluppato 4 comandi per controllare l'apparato meccanico MOE tramite riferimenti di posizione e di velocità. Vengono selezionati valorizzando il Byte id del pacchetto msg_rcvd con il codice comando corrispondente. Per ogni pacchetto msg_rcvd inviato al PIC è possibile richiamare un solo comando. I comandi sono:

get : codice comando: 0x11. Non bloccante rispetto a un comando di set. Con tale comando il PIC risponderà inviando il messaggio msg_sent con la posizione attuale delle 2 telecamere. Tale posizione si riferisce al momento dell'invio del messaggio msg_sent da parte del PIC; attualmente il sistema non fornisce nessun meccanismo per controllarne l'effettiva coerenza successiva all'istante di invio del pacchetto. Un sistema per ottenere delle informazioni aggiornate sullo stato del sistema consiste quindi nel fare polling con comandi di get. I campi, escluso l'id, in un messaggio di get sono ininfluenti (ricordiamo però che la checksum deve essere coerente con quanto inviato).

set : codice comando: 0x22. Sovrascrive un eventuale set in corso. Il PIC inizierà a muovere gli attuatori per raggiungere la posizione richiesta, mantenendo la velocità richiesta. Arrivato nella posizione designata non notificherà l'evento al PC. Un sistema per ottenere delle informazioni aggiornate sullo stato del sistema consiste quindi nel fare polling con comandi di get. I campi, escluso l'id, in un messaggio di stop sono ininfluenti (ricordiamo però che la checksum deve essere coerente con quanto inviato).

stop : codice comando: 0x33. Bloccante rispetto al set. Interrompe un eventuale movimento in corso.

reset : codice comando: 0x44. Bloccante rispetto al set. Muove i motori, mantenendo la velocità richiesta, fino al fine corsa destro di ciascuno di essi, riportandoli poi nella posizione iniziale designata da noi al momento della taratura. Finché il movimento di reset non viene completato ogni altro comando viene messo in coda, ed eseguito al completamento del movimento. I campi, escluso l'id e le velocità dei singoli motori, in un messaggio di reset sono ininfluenti (ricordiamo però che la checksum deve essere coerente con quanto inviato).

7.5 Gestione dello stato

Il firmware si comporta come una macchina a stati. Ne illustriamo la dinamica tramite un diagramma degli stati relativo al movimento di un solo motore. Gli stati del sistema sono i seguenti:

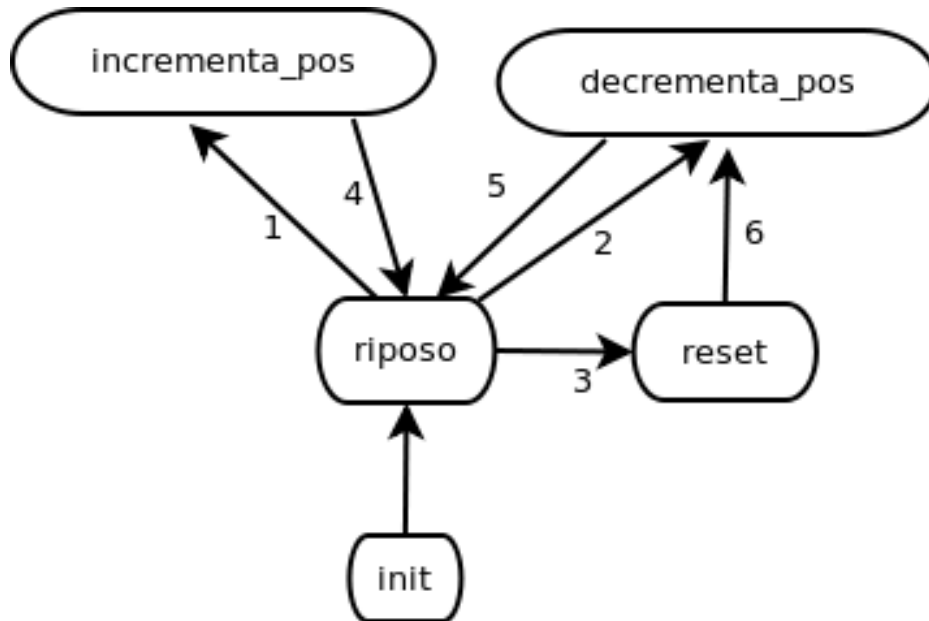


Figura 7.5: Diagramma gestione dello stato

init : inizializza le variabili di stato;

riposo : motore fermo;

incrementa_pos : incremento la posizione, mantenendo la velocità di riferimento. Il motore è in movimento. Raggiunto il riferimento passo allo stato riposo;

decrementa_pos : decremento la posizione, mantenendo la velocità di riferimento. Il motore è in movimento. Raggiunto il riferimento passo allo stato riposo;

reset : esegue movimento di reset. Arrivato a fine corsa passo allo stato decrementa_pos, settando un riferimento fisso (impostato da taratura).

Le variabili di stato del sistema sono le seguenti:

movimento : se valorizzata a '1' il motore è in movimento;

pos_y : posizione motore y;

spd_y : velocità motore y;

reset : se valorizzata a '1' il motore sta effettuando il movimento di reset.

La posizione di un motore è un intero compreso tra 0 e 255. La posizione iniziale all'accensione del sistema è 128. Ogni unità rappresenta una variazione di 0.9° dalla posizione iniziale (che indichiamo 0°). Per convenzione all'aumentare della variabile di posizione corrisponde un movimento orario. Simmetricamente abbiamo che al diminuire della variabile corrisponde un movimento antiorario. Ad esempio, una posizione di $+9^\circ$ rispetto allo 0° verrebbe interpretata dal sistema come 138, mentre una posizione di -9° 118. Il riferimento (ovvero la posizione richiesta) che il PC invia al PIC segue la stessa convenzione.

Per il controllo di velocità del movimento è stato necessario introdurre una routine temporizzata. Ogni step (sia half che full) si ottiene tramite un fronte di salita nel PIN di clock di ingresso del L297. Temporizzando tale fronte di salita è possibile temporizzare il movimento di ogni singolo step. Il valore di velocità inviato dal PC al PIC è un intero compreso tra 0 e 255. Il firmware temporizza l'inversione della tensione TTL nel PIN di clock del L297. Per fare questo conta quanti interrupt sono trascorsi dall'ultimo cambio di fronte. Se tale valore è minore o uguale al valore inviato come velocità richiesta inverte lo stato. C'è quindi una corrispondenza biunivoca tra il valore speed inviato $[0,255]$ e il valore di velocità reale (stimato) dimensionato in grad/sec.

Se chiamo $T[\text{sec}]$ il periodo di interrupt e spd il valore di velocità inviato, ottengo un cambio di fronte ogni $T \cdot \text{speed}[\text{sec}]$. Siccome L297 pilota uno step per ogni fronte di salita del clock, avremo uno step ogni $2 \cdot T \cdot \text{speed}[\text{sec}]$. Essendo ogni full step di 1.8° e $T = 130.5 \mu\text{s}$, invertendo la precedente formula, otteniamo i valori riportati in 7.3.

Nella tabella 7.3 sono portati i primi 10 valori del campo SPD nel pacchetto `msg_rcvd`, nel caso del full step. Se il sistema si muove in half step le velocità vanno divise per due.

Il controllo in posizione viene fatto ad anello aperto, così come quello in velocità. Ad ogni fronte di salita del PIN di clock del L297 il firmware aggiorna la posizione di ± 2 unità (quindi $\pm 1.8^\circ$) se si muove in full step; oppure ± 1 unità (quindi $\pm 0.9^\circ$) se in half step. Esiste tuttavia un'eccezione: durante un movimento full step, come ultimo movimento il sistema può

velocità in full[grad/sec]	SPD
5984	0
2992	1
1995	2
1496	3
1197	4
997	5
855	6
748	7
665	8
598	9
544	10

Tabella 7.3: Estratto della tabella di velocità

forzare, se necessario, un half step per ottenere maggiore precisione. Al fine di chiarire tale procedura descriviamo un possibile esempio di funzionamento del sistema: posizione attuale = 128; posizione richiesta = 139; tipo di movimento = full step. Il sistema farà 5 movimenti in full step portandosi alla posizione 138, e concluderà con un singolo half step per portarsi alle 139. Nel caso di movimento half step chiaramente non sussistono eccezioni in quanto la risoluzione del movimento è già minima.

Il movimento di reset consiste in un movimento orario senza controllo di riferimento. Quando il registro corrisponde al PIN di input del sensore ottico si abbassa, significa che il movimento è arrivato al suo finecorsa. Immediatamente il sistema effettua un movimento antiorario (di 35 gradi per i motori orizzontali e 50 gradi per il verticale) per tornare nello stato (0,0,0).

7.5.1 Interrupt nel PIC16F877

L'interrupt è una particolare caratteristica dei microprocessori che consente di intercettare un evento esterno, interrompere momentaneamente il programma in corso, eseguire una porzione di programma specializzata per la gestione dell'evento verificatosi e riprendere l'esecuzione del programma principale. E' utile infatti che certi eventi particolari vengano rilevati in un qualsiasi momento dell'esecuzione del programma, come ad esempio l'arrivo di una comunicazione seriale, l'overflow di un timer, oppure un fronte di salita di un pin. In questi casi entrano in gioco gli interrupt: ad un preciso evento

richiesto, il controllore interrompe l'esecuzione del programma principale e passa ad una *subroutine di interrupt*. Nel caso della seriale, tale subroutine si incaricherà di ricevere correttamente il byte trasmesso dall'esterno. Al termine della routine di interrupt il programma riprende dal punto in cui aveva interrotto l'esecuzione.

Il PIC16F877 ha fino a 14 sorgenti di interrupt, ai fini della stesura del firmware, abbiamo scelto di temporizzare la gestione dello stato e il pilotaggio dell'integrato L297 utilizzando la fine del conteggio del registro TMR0 (TMR0 Overflow Interrupt). Qualunque sia l'evento abilitato, al suo manifestarsi il PIC interrompe l'esecuzione del programma in corso, memorizza automaticamente nello STACK il valore corrente del PROGRAM COUNTER e salta all'istruzione presente nella locazione di memoria denominata Interrupt Vector. A questo punto è necessario inserire nel codice la subroutine di gestione degli interrupt denominata Interrupt Handler. Avendo la possibilità di abilitare più interrupt, uno dei primi compiti dell'interrupt handler è la verifica di quale evento, tra gli eventi abilitati, ha generato l'interrupt e l'esecuzione nella parte di programma relativo. Questo controllo può essere effettuato utilizzando gli interrupt flag nel registro INTCON.

Per abilitare l'interrupt TMR0 Overflow Interrupt va impostato a '1' il bit 2 del registro INTCON.

Per dimensionare il tempo di interrupt TMR0, a partire dal TOSC (nel nostro caso 20Mhz) agiamo sul prescaler ed impostiamo un offset sul registro del contatore stesso.

```

1 // Impostazione del registro OPTION (pag.55 del datasheet)
2 OPTION=0b11000001;
3 // bit 0 -> Prescaler Rate Select bit 0
4 // bit 1 -> Prescaler Rate Select bit 0
5 // bit 2 -> Prescaler Rate Select bit 0 (1:32)
6 // bit 3 -> Prescaler assegnato al Timer0
7 // bit 4 -> Non importa
8 // bit 5 -> Clock per Timer0 derivato da ciclo di clock
   interno
9 // bit 6 -> Non importa
10 // bit 7 -> Resistenze di pull-up su porta B disattivate
11
12 // Impostazione Interrupt
13 INTCON=0b10100000;
14 // bit 0 -> RBIF - Flag interrupt su porte B
15 // bit 1 -> INTF - Flag interrupt su RB0/INT
16 // bit 2 -> T0IF - Flag interrupt su Timer0
17 // bit 3 -> RBIE, Interrupt su porte B disattivato
18 // bit 4 -> INTE, Interrupt su porta RB0/INT disattivato

```

```
19 // bit 5 -> TMR0IE, Interrupt su Timer0 attivato
20 // bit 6 -> PEIE, Interrupt di periferica disattivato
21 // bit 7 -> GIE, Gestione Interrupt attiva
22
23 TMR0=68;
```

Per quanto concerne la tesi in esame, abbiamo impostato il periodo di interrupt a 130,5 μ s, circa 3 volte il tempo massimo di esecuzione della routine di interrupt. Programmaticamente la routine di interrupt appare nel seguente modo:

```
1 if (T0IF) {
2     /* logica da eseguire */
3     TMR0 = 68;
4     T0IF = 0;
5 }
```

Il registro T0IF viene valorizzato dal PIC a '1' quando il timer va in overflow. Il registro TMR0 contiene il valore del contatore incrementato una volta ogni 130,5 μ s.

INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

- bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts
- bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
- bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt
- bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt
- bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt
- bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow
- bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur
- bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figura 7.6: Registro TOIF

Capitolo 8

Libreria MOE

“Il C rende facile spararsi in un gamba; il C++ lo rende più difficile, ma ti fa saltare in aria tutta la gamba.”

Bjarne Stroustrup (1950), padre del c++ ...

In questo capitolo mostreremo una libreria di funzioni implementata per la realizzazione di routine di controllo dell'apparato MOE. Tale libreria, statica per sistema operativo linux, prende il nome di *libmoe*. Concluderemo il capitolo mostrando come realizzare una semplice interfaccia di controllo dell'apparato utilizzando *libmoe*.

8.1 Introduzione

L'ultimo passo nello sviluppo della nostra tesi consiste nella realizzazione di una libreria codificata in C, contenente l'insieme iniziale di primitive per il controllo dell'apparato. Con tali primitive uno sviluppatore può progettare facilmente una routine di controllo per il sistema; allo stesso tempo potrà essere altrettanto facilmente espansa e migliorata da chiunque vorrà cimentarvisi in futuro.

8.2 Libreria

Una libreria è semplicemente un file contenente codice compilato che può essere successivamente incorporato come una unica entità in un nostro programma in fase di linking; l'utilizzo delle librerie ci permette di realizzare programmi più facili da compilare e mantenere. Di norma le librerie sono indicizzate, in modo da rendere più facile localizzare simboli (funzioni, variabili, classi, etc...) al loro interno. Per questa ragione il link ad una libreria è

più veloce rispetto al caso in cui i moduli oggetto siano separati nel disco. Inoltre, quando usiamo una libreria abbiamo meno file da aprire e controllare, e questo comporta un ulteriore aumento della velocità del processo di link.

Nell'ambiente Linux (come nella maggior parte dei sistemi moderni) le librerie si suddividono in due famiglie principali:

- librerie statiche (static libraries);
- librerie dinamiche o condivise (shared libraries);

Le librerie statiche vengono installate nell'eseguibile del programma prima che questo possa essere lanciato. Esse sono semplicemente cataloghi di moduli oggetto che sono stati collezionati in un unico file contenitore. Le librerie statiche ci permettono di effettuare dei link di programmi senza dover ricompilare il loro codice sorgente. Per far girare il nostro programma abbiamo bisogno solo del suo file eseguibile.

Le librerie condivise (dette anche dinamiche) vengono collegate ad un programma in due passaggi. In un primo momento, durante la fase di compilazione (Compile Time), il linker verifica che tutti i simboli (funzioni, variabili, classi, e simili ...) richiesti dal programma siano effettivamente collegati al programma o ad una delle sue librerie condivise. In ogni caso i moduli oggetto della libreria dinamica non vengono inseriti direttamente nel file eseguibile. In un secondo momento, quando l'eseguibile viene lanciato (Run Time), un programma di sistema (dynamic loader) controlla quali librerie dinamiche siano state collegate al nostro programma, le carica in memoria e le collega alla copia del programma in memoria.

8.3 libmoe e tabelle di velocità

La libreria *libmoe* è una libreria statica sviluppata utilizzando programmi facenti parte della gnu toolchain quali il compilatore gcc e il programma gnu make. Essa si compone di tre parti: il parser csv, le funzioni di utilità, e i comandi verso il firmware MOE.

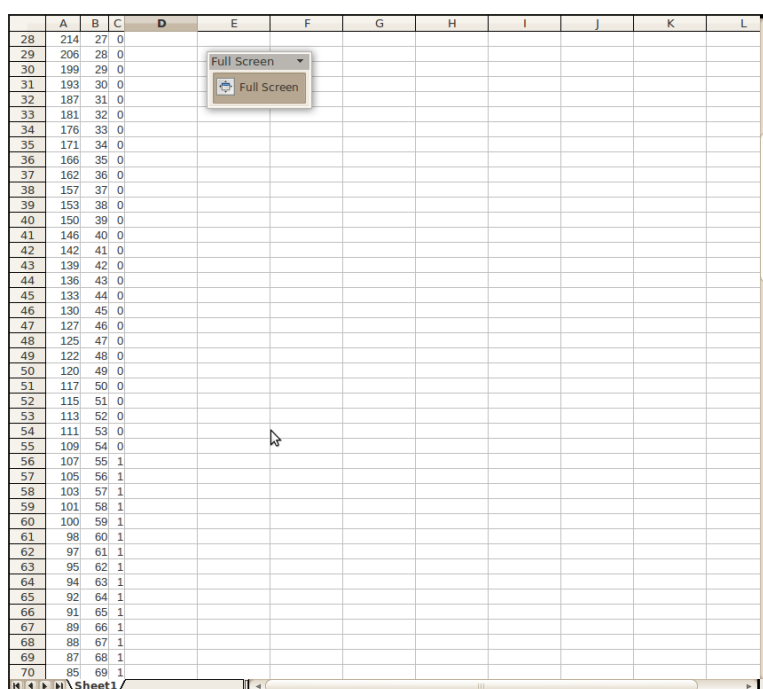
La scelta di un parser per il formato csv non è banale. Un comma-separated values (abbreviato in CSV) è un formato basato su file di testo utilizzato per l'importazione ed esportazione (ad esempio da fogli elettronici o database) di una tabella di dati. Non esiste uno standard formale che lo definisca, ma solo alcune prassi più o meno consolidate. *libmoe* considera come csv un formato del tipo:

1 num1 ; num2 ; num3 \n

dove num1,num2 sono due interi positivi, mentre num3 può assumere valori '0' o '1'.

Tramite questo formato l'utente può comunicare a *libmoe* una tabella delle velocità. Tale tabella descrive un profilo di velocità raggiungibili da un motore dell'apparato MOE, vi associa il valore corrispondente della proprietà SPD utilizzata dal firmware per il controllo in velocità e un flag di enable. Durante i nostri test infatti ci è capitato di testare delle velocità che il motore non riusciva a mantenere senza perdere il passo. Ci è sembrato quindi essenziale fornire allo sviluppatore un modo semplice e veloce per modificare on-the-fly il profilo del proprio motore senza dover -ad esempio- ricompilare il progetto. L'unico vincolo è di tenere una copia delle tabelle nello stesso path dell'eseguibile implementato.

Le tabelle sono due: una per il motore che muove il carico verticalmente, una per i due motori orizzontali (che si presume siano uguali). Il contenuto di tale tabella è associato al periodo di interrupt scelto lato firmware. Segue un esempio di tabella di velocità utilizzata.



	A	B	C
28	214	27	0
29	206	28	0
30	199	29	0
31	193	30	0
32	187	31	0
33	181	32	0
34	176	33	0
35	171	34	0
36	166	35	0
37	162	36	0
38	157	37	0
39	153	38	0
40	150	39	0
41	146	40	0
42	142	41	0
43	139	42	0
44	136	43	0
45	133	44	0
46	130	45	0
47	127	46	0
48	125	47	0
49	122	48	0
50	120	49	0
51	117	50	0
52	115	51	0
53	113	52	0
54	111	53	0
55	109	54	0
56	107	55	1
57	105	56	1
58	103	57	1
59	101	58	1
60	100	59	1
61	98	60	1
62	97	61	1
63	95	62	1
64	94	63	1
65	92	64	1
66	91	65	1
67	89	66	1
68	88	67	1
69	87	68	1
70	85	69	1

Figura 8.1: Esempio di tabella di velocità

Nella figura 8.1 notiamo che la tabella di velocità è composta dalle 3 colonne descritte in precedenza. In particolare la prima colonna indica la velocità di rotazione del motore espressa in grad/sec, la seconda indica il

valore intero da inviare al PIC, la terza è il flag di enable. In questo caso la prima velocità disponibile corrisponde alla tupla di valori alla riga 56, ovvero 107 grad/sec a cui associamo il valore 55 e il flag di enable a 1.

8.3.1 Makefile e grafo dipendenze

La libreria MOE è composta da 3 oggetti: `parse_csv`, `utils` e `command`. Illustriamo le dipendenze degli oggetti in questione con il grafo in figura 8.2.

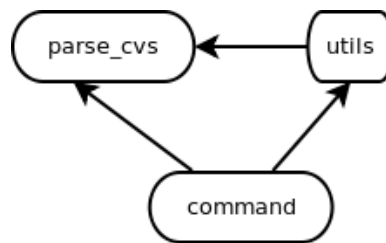


Figura 8.2: Grafo dipendenze oggetti libmoe

L'oggetto `command` dipende da `utils` e `parse_csv`. L'oggetto `utils` dipende solo da `parse_csv`. Mentre l'oggetto `parse_csv` non ha dipendenze. Un possibile makefile che rispetti tale grafo è il seguente:

```
1 all: parse_csv.o utils.o command.o test.c
2     gcc -lm -o test test.c parse_csv.o utils.o command.
3     o
4 parse_csv.o: parse_csv.c parse_csv.h
5     gcc -c -lm parse_csv.c
6
7 utils.o: parse_csv.c parse_csv.h utils.c utils.h
8     gcc -c -lm utils.c
9
10 command.o: parse_csv.c parse_csv.h utils.c utils.h command.
11     c command.h
12     gcc -c -lm command.c
```

8.4 MOEtest

In questa sezione descriveremo la realizzazione di MOEtest, ovvero di un sistema di controllo manuale per l'apparato MOE. Lo scopo di questo progetto è di illustrare al lettore un esempio di utilizzo della libreria *libmoe*.

L'idea alla base del MOEtest è quella di consentire il controllo delle webcam all'utente tramite il touchpad di un portatile. Tale controllo viene effettuato sfiorando l'area sensibile del touchpad con un solo dito. Il centro del touchpad corrisponde alla posizione di reset (0,0,0) mentre le estremità orizzontali e verticali corrispondono ai fine corsa verticale e orizzontali rispettivamente dei 3 motori. In aggiunta vogliamo inviare un comando di reset in risposta a un evento di left click.

Per ottenere questo controllo abbiamo bisogno di intercettare l'evento di sfioramento del touchpad, in posizione assoluta nell'area sensibile del dispositivo, nonché l'evento di pressione del tasto sinistro. Nella maggioranza dei portatili odierni i touchpad integrati sono realizzati dalla Synaptic, la quale ha rilasciato dei driver opensource reperibili facilmente in rete. La Synaptic fornisce anche un programma, *synclient*, preinstallato in molte distribuzioni linux, che tra le altre cose stampa nello standard output gli eventi che interessano il touchpad. Decidiamo quindi di procedere implementando un parser che riceva come input le informazioni stampate da *synclient* e ricavi da esse gli eventi di sfioramento e pressione tasto sinistro.

8.4.1 Synclient

Il programma *synclient* lanciato con opportuni parametri si comporta da demone intercettando gli eventi del touchpad e stampandoli nello standard output. Se chiamiamo da terminale il comando `man synclient` otteniamo le seguenti informazioni:

NAME *synclient* - commandline utility to query and modify Synaptics driver options.

SYNOPSIS *synclient* [-m interval] *synclient* [-hlV?] [var1=value1] ...

DESCRIPTION This program lets you change your Synaptics TouchPad driver for XOrg/XFree86 server parameters while X is running. For the -m and -h options, SHM must be enabled by setting the option SHMConfig on in your XOrg/XFree86 configuration.

OPTIONS **-m interval** monitor changes to the touchpad state. Interval specifies how often (in ms) to poll the touchpad state. Whenever

a change in the touchpad state is detected, one line of output is generated that describes the current state of the touchpad. This option is only available in SHM mode. The following data is included in the output.

time Time in seconds since the logging was started.

x,y The x/y coordinates of the finger on the touchpad. The origin is in the upper left corner.

z The pressure value. Pressing the finger harder on the touchpad typically produces a larger value. Note that most touchpads don't measure the real pressure though. Instead, capacitance is usually measured, which is correlated to the contact area between the finger and the touchpad. Since more pressure usually means a larger contact area, the reported pressure value is at least indirectly related to the real pressure.

f The number of fingers currently touching the touchpad. Note that only some touchpads can report more than one finger. Generally, synaptics touchpads can, but ALPS touchpads can't.

w The w value is a measurement of the finger width. This is only supported by some synaptics touchpads. Touchpads that can't measure the finger width typically report a faked constant value when a finger is touching the touchpad.

l,r,u,d,m,multi The state of the left, right, up, down, middle and multi buttons. Zero means not pressed, one means pressed. Not all touchpads have all these buttons. If a button doesn't exist, the value is always reported as 0.

gl,gm,gr Some touchpads have a guest device. This is typically a pointing stick located in the middle of the keyboard. Some guest devices have physical buttons, or can detect button presses when tapping on the pointing stick. Such button events are reported as guest left, guest middle or guest right.

gdx,gdy Pointer movements from the guest device are reported as relative x/y positions, called gdx and gdy.

-l List current user settings.

-V Print version number and exit.

-? Show the help message.

var=value Set user parameter var to value.

L'entry nel man ci comunica che il `synclient -m` è invocabile in modalità SHM. In tale modalità il driver synaptic scrive in una shared memory le informazioni sullo stato del touchpad. Per abilitare tale modalità cerchiamo la entry del device nel file di configurazione `xorg.conf`, e assicuriamoci che si presenti in questo modo (se apportiamo modifiche dobbiamo riavviare x server), come riportato nell'articolo [8]:

```

1 Section                "InputDevice"
2   Identifier            "Synaptics_Touchpad"
3   Driver                 "synaptics"
4   Option                 "SHMConfig"          "on"
5 EndSection

```

E' importante verificare che la entry Option SHMConfig sia impostata su on.

A questo punto se invochiamo da terminale il comando con tempo di campionamento 100 ms

```

1 synclient -m 100

```

otteniamo un output simile a quello presente nella figura 8.3:

```

renzo@renzo-laptop:~$ synclient -m 100
time    x    y    z  f  w  l  r  u  d  m    multi  gl  gm  gr  gdx  gdy
0.000  1108 3657  0  0  0  0  0  0  0  0  00000000  0  0  0  0  0
1.001  4527 2256 60  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.101  4094 2390 64  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.201  3809 2883 65  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.302  3946 3058 71  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.402  4387 2976 72  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.502  4581 2231 67  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.602  4177 1743 71  1  5  0  0  0  0  0  00000000  0  0  0  0  0
1.702  3507 1856 67  1  5  0  0  0  0  0  00000000  0  0  0  0  0
1.802  3164 2417 73  1  4  0  0  0  0  0  00000000  0  0  0  0  0
1.902  3334 3154 71  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.003  3780 3481 77  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.103  4135 3160 77  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.203  4492 2497 75  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.303  4430 1994 71  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.403  3992 1781 72  1  5  0  0  0  0  0  00000000  0  0  0  0  0
2.503  3592 1981 66  1  5  0  0  0  0  0  00000000  0  0  0  0  0
2.603  3457 2247 73  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.704  3675 2536 66  1  4  0  0  0  0  0  00000000  0  0  0  0  0
2.804  3785 2271  0  0  0  0  0  0  0  0  00000000  0  0  0  0  0

```

Figura 8.3: Output `synclient`

Le informazioni di interesse sono il timestamp, il valore x, il valore y e il flag di bottone sinistro premuto (l). Per passare i dati al nostro programma possiamo utilizzare una pipe, direttamente da terminale:

```
1 synclient -m 100 | nostro_programma
```

In questo modo possiamo fare il parsing delle informazioni direttamente dallo standard input.

8.4.2 Utilizzare libmoe

Illustriamo di seguito come realizzare un controller per l'apparato MOE utilizzando la libreria *libmoe*. Come prima cosa compiliamo la libreria e la installiamo nel nostro sistema. Dalla cartella dei sorgenti della libreria eseguiamo nel terminale:

```
1 make && sudo make install
```

In questo modo compileremo gli oggetti di cui *libmoe* è composta, li archiveremo in formato ar nel file *libmoe.a* e copieremo tale file nella cartella */usr/lib*.

Il controller che chiamiamo *test* sarà composto da un solo sorgente. Illustriamo un possibile esempio di programma.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include "command.h"
5
6 #define LEFT_EDGE 970
7 #define RIGHT_EDGE 6000
8 #define TOP_EDGE 940
9 #define BOTTOM_EDGE 4470
10 #define DELTA_Y_MOE 60
11 #define DELTA_X_MOE 90
12 #define FILTER_TRESHOLD_X 60
13 #define FILTER_TRESHOLD_Y 60
14 #define PRESSURE_TRESHOLD 60
15
16 #define T_CAMP 0.1 //[sec]
17
18 #define DELAY 1
19
20 int main()
21 {
22     double time = 0;
```

```
23     float time_old = 0;
24     int x = 0;
25     int y = 0;
26     int x_old = 0;
27     int y_old = 0;
28
29     int i = 0;
30
31     char temp_string[512];
32     char unused;
33
34     if(!MOE_open("speed_table_x.csv","speed_table_y.csv
35                 "))
36     {
37         return 0;
38     }
39     while(1)
40     {
41         while(1)
42         {
43             scanf("%s\n",temp_string);
44             if(!strcmp(temp_string,"gdy"))
45             {
46                 break;
47             }
48         }
49         for(i=0;i<20;i++)
50         {
51             int reset = 0;
52             int pressure = 0;
53             int discard;
54             scanf("%lf %d %d %d %d %d %d",&time,&x,&
55                 y,&pressure,&discard,&discard,&discard,&
56                 reset);
57             while(1)
58             {
59                 unused=getchar();
60                 if(unused == '\n')
61                     break;
62             }
63
64             if(time == 0)
65                 continue;
```

```

65         if(reset == 1)
66         {
67             MOE_reset(650);
68             continue;
69         }
70
71         if(pressure <=
72             PRESSURE_TRESHOLD)
73             continue;
74
75         unsigned char event_rcvd = 0;
76
77         if(abs(x-x_old) > FILTER_TRESHOLD_X)
78             event_rcvd = 1;
79
80         if(abs(y-y_old) > FILTER_TRESHOLD_Y)
81             event_rcvd = 1;
82
83         if (event_rcvd)
84         {
85             double max_range_tracking_x = 0;
86             double max_range_tracking_y = 0;
87
88             if(tracking_delay != 0)
89             {
90                 tracking_delay--;
91                 continue;
92             }
93
94             if(x_old == 0 && y_old == 0 && time_old
95                == 0)
96             {
97                 x_old = x;
98                 y_old = y;
99                 time_old = time;
100             }
101
102             double avg_speed = 0;
103             avg_speed = sqrt( pow(abs(x-x_old) *
104                 DELTA_X_MOE / (RIGHT_EDGE -
105                 LEFT_EDGE) / T_CAMP, 2) + pow(abs(y-
106                 y_old) * DELTA_Y_MOE / (BOTTOM_EDGE
107                 - TOP_EDGE) / T_CAMP, 2));

```

```

103         double pos_x = (x - LEFT_EDGE) *
104             DELTA_X_MOE / (RIGHT_EDGE -
105             LEFT_EDGE) - DELTA_X_MOE/2;
106             double pos_y = (y -
107                 TOP_EDGE) *
108                 DELTA_Y_MOE / (
109                 BOTTOM_EDGE -
110                 TOP_EDGE) -
111                 DELTA_Y_MOE/2;
112
113         MOE_set(pos_y, pos_x, pos_x, 650);
114
115         x_old = x;
116         y_old = y;
117         time_old = time;
118     }
119 }
120 }
121 }

```

La documentazione dei comandi di *libmoe* può essere reperita nella sezione A.2. Come prima cosa notiamo l'utilizzo della funzione `MOE_open`, necessaria per inizializzare il collegamento seriale con l'apparato. Quello che segue è il parsing dello standard input, che proviene dal `synclient`. I dati di nostro interesse sono il timestamp, i riferimenti `x` e `y`, la pressione e il flag indicante la pressione del left button del touchpad. Il resto delle informazioni viene scartato. Ulteriore elemento di difficoltà è costituito dall'header che `synclient` invia (ogni 20 eventi stampati).

In programma implementato ci si cerca l'header, e dopo averlo riconosciuto e scartato si aspetta 20 eventi validi, dopodiché ancora un header e così via. Per rendere il controller utilizzabile abbiamo previsto dei semplici filtri software, ovvero un filtro sulla pressione esercitata sul touchpad e sulla variazione di posizione. In questo modo scartiamo eventi con un valore di pressione basso (ovvero sfioramenti non voluti del touchpad) oppure variazioni minime dall'ultima posizione registrata (rumore bianco del dispositivo). Un ulteriore filtro consiste nell'inibire un evento se il tempo intercorso tra l'evento e l'ultimo registrato sia minore di una certa soglia.

I filtri sopra descritti sono impostabili nelle label in testa al listato.

Notiamo l'utilizzo del comando `MOE_set`. I parametri passati al comando sono i riferimenti di posizione ottenuti tramite touchpad, proporzionati nello

spazio visivo dell'apparato MOE. La velocità richiesta è fissa a 650 grad/sec. In questo modo il centro del touchpad corrisponde allo stato (0,0,0) delle webcam.

Se riceviamo l'evento di pressione del tasto inviamo il comando di reset `MOE_reset` con velocità 650 grad/sec.

Per compilare il test descritto occorre copiare nella cartella del sorgente le tabelle di velocità e gli header della libreria. Dopodiché possiamo usare il comando da terminale:

```
1 gcc -o test test.c -lmoe
```

Capitolo 9

Valutazioni sperimentali MOE

*“La scienza è fatta di dati come una casa di pietre.
Ma un ammasso di dati non è scienza più di quanto un mucchio di pietre sia
una casa.”*

Jules Henri Poincaré (1854-1912), matematico francese ...

In questa sezione mostreremo alcuni test effettuati sul nostro sistema, come il test sulle posizioni (sezione 9.1), sulle velocità (sezione 9.2) e sulla precisione della rotazione (sezione 9.3).

9.1 Test sulle posizioni

Il nostro sistema è stato inizialmente testato per verificare che ogni singola posizione di ogni motore fosse funzionante. Per eseguire il test sono stati utilizzati degli script che eseguivano la seguente procedura in pseudo codice:

```
1 for (motore==1; motore<=3; motore++)  
2 {  
3   for (   posizione==0;  
4         posizione<=posizione_fine_corsa;  
5         posizione+=0.9)  
6   {  
7     test [motore] [ posizione ];  
8     sleep [1];  
9     test [motore] [ posizione-posizione_fine_corsa ];  
10    sleep [1];  
11  }  
12 }
```

In questo modo abbiamo controllato tutte le posizioni dallo zero al fine corsa che ricordiamo essere di ± 30 gradi per il motore verticale e di ± 45 gradi per i due motori orizzontali.

9.2 Test di velocità

I test di velocità sono stati eseguiti in due fasi: la prima fase consisteva nella verifica delle velocità raggiungibili dai motori orizzontali, questi infatti presentavano una struttura simmetrica strutturalmente identica ed abbiamo supposto che raggiungessero le stesse velocità. Abbiamo quindi testato, via software, tutte le velocità ed abbiamo compilato una tabella in cui mostravamo quali velocità non fossero raggiungibili. Per questa prima fase abbiamo rilevato una velocità massima raggiungibile per i due motori sull'asse orizzontale, su spostamenti di circa 40° , pari a $665^\circ/\text{s}$. Per velocità superiori i motori non eseguivano lo spostamento correttamente dato che perdevano sistematicamente il passo. Nonostante ciò la massima velocità raggiungibile dai piccoli motori adibiti al movimento orizzontale è soddisfacente per emulare i movimenti saccadici umani.

Nella seconda fase dei test abbiamo analizzato il comportamento del primo motore e cioè quello che muove la struttura portante degli altri due motori. In questa situazione il carico è notevolmente maggiore ed è pari a circa 600 grammi connessi direttamente all'albero del motore. Nonostante il carico sia collegato saldamente all'albero e uniforme sia la distribuzione del peso (data la configurazione asimmetrica dei due motori), la struttura ed i motori soprattutto sono stati sempre un punto critico di MOE a causa dell'eccessivo peso. Il motore verticale, come previsto, non è riuscito a raggiungere le velocità dei due motori orizzontali nonostante fosse notevolmente più potente. Durante i test effettuati la velocità massima raggiungibile rilevata, per spostamenti di circa 40° , è stata pari a $214^\circ/\text{s}$. Ciò nonostante questa velocità è stata calcolata su angoli di spostamento relativamente ampi; ricordiamo che come spiegato nel capitolo dedicato alla Bioispirazione nella sezione 4, normalmente una saccade si protrae per circa 20° salvo poi continuare il movimento di inseguimento veloce con l'ausilio del collo. Inoltre come spiegheremo nel capitolo 10 riguardante gli sviluppi futuri, nel caso si disponesse di maggiori risorse, sarebbe possibile sostituire il motore numero 1 responsabile della minore velocità massima raggiungibile con un nuovo motore più performante. Abbiamo poi calcolato l'effettiva velocità risultante dal moto combinato del motore verticale e dei motori orizzontali ottenendo la tabella 9.1 indicante le velocità raggiungibili in base alle diverse angolazioni del movimento rispetto all'orizzonte. Tali velocità massime risultano talvolta insufficienti per l'emu-

lazione dei movimenti saccadici per spostamenti superiori ai 5° . Tuttavia, nella maggioranza dei movimenti possibili, il sistema soddisfa le richieste.

Angolo rispetto all'orizzonte	Rotazione massima raggiungibile	Adeguito per saccadi di		
		5° ($150^\circ/\text{s}$) ?	10° ($300^\circ/\text{s}$) ?	30° ($500^\circ/\text{s}$) ?
0°	$665^\circ/\text{s}$	si	si	si
5°	$681^\circ/\text{s}$	si	si	si
10°	$692^\circ/\text{s}$	si	si	si
15°	$698^\circ/\text{s}$	si	si	si
20°	$698^\circ/\text{s}$	si	si	si
25°	$693^\circ/\text{s}$	si	si	si
30°	$683^\circ/\text{s}$	si	si	si
35°	$667^\circ/\text{s}$	si	si	si
40°	$647^\circ/\text{s}$	si	si	si
45°	$622^\circ/\text{s}$	si	si	si
50°	$591^\circ/\text{s}$	si	si	si
55°	$557^\circ/\text{s}$	si	si	si
60°	$518^\circ/\text{s}$	si	si	si
65°	$475^\circ/\text{s}$	si	si	NO
70°	$429^\circ/\text{s}$	si	si	NO
75°	$379^\circ/\text{s}$	si	si	NO
80°	$326^\circ/\text{s}$	si	si	NO
85°	$271^\circ/\text{s}$	si	NO	NO
90°	$214^\circ/\text{s}$	si	NO	NO

Tabella 9.1: Velocità di rotazione raggiungibili in base all'angolo di spostamento

9.3 Test di precisione

Il passo dei motori utilizzati è di 1.8° , il mezzo passo è di 0.9° . In questa condizione l'errore dello spostamento dipende fortemente dall'entità dello spostamento, dato che è una composizione di errore di risoluzione ed errore sistematico. L'errore di risoluzione è provocato dal fatto che il motore si muove di spostamenti multipli di 0.9° . Per questo motivo se l'angolo di spostamento non è multiplo di 0.9° il motore si andrà a posizionare nello stato più vicino disponibile. L'errore di risoluzione è quindi pari a zero in tutti gli stati multipli di 0.9° , mentre massimo e pari a 0.45° nel caso di posizione richiesta perfettamente centrata tra due stati multipli di 0.9° ad esempio

0.45° appunto. L'errore sistematico invece è quello che porta la struttura ad effettuare spostamenti di entità differente rispetto a quella fornita a causa di imprecisioni meccaniche o giochi. Abbiamo deciso quindi di effettuare dei test per verificare l'errore sistematico in ogni possibile situazione di pilotaggio del motore. I risultati dei test effettuati sono esposti nella tabella 9.2:

Test #	Motore	Cam	Drive	Step	Passo di
Test 1	1	Right	Normal	Full	1.8°
Test 2	1	Right	Wave	Full	1.8°
Test 3	1	Right	Both	Half	0.9°
Test 4	1	Right	Normal	Full	5.4°
Test 5	1	Right	Wave	Full	5.4°
Test 6	1	Right	Both	Half	4.5°
Test 7	2	Left	Normal	Full	1.8°
Test 8	2	Left	Wave	Full	1.8°
Test 9	2	Left	Both	Half	0.9°
Test 10	2	Left	Normal	Full	5.4°
Test 11	2	Left	Wave	Full	5.4°
Test 12	2	Left	Both	Half	4.5°
Test 13	3	Right	Normal	Full	1.8°
Test 14	3	Right	Wave	Full	1.8°
Test 15	3	Right	Both	Half	0.9°
Test 16	3	Right	Normal	Full	5.4°
Test 17	3	Right	Wave	Full	5.4°
Test 18	3	Right	Both	Half	4.5°

Tabella 9.2: Test eseguiti

Abbiamo effettuato diciotto differenti test di precisione sul sistema motore-webcam. Si è reso necessario un numero così elevato di prove per determinare la precisione di ogni singolo motore in ogni possibile tipologia di movimento dei motori passo passo. Ogni motore quindi è stato sottoposto a sei differenti test, i primi tre prevedevano una serie di tredici movimenti molto brevi; gli ultimi tre invece prevedevano cinque o sei movimenti di medio angolo. Non abbiamo ritenuto necessario valutare la precisione nel caso di movimenti lunghi per due motivi: prima di tutto tali risultati possono essere estrapolati facilmente dai test eseguiti sui piccoli movimenti; in secondo luogo per la natura dei test era difficile la valutazione di angoli maggiori di 25°. Per superare questo problema sarebbe stato necessario sostituire la scacchiera utilizzata per i test e disporla a diversi metri dalle telecamere.

9.3.1 Raccolta dei dati

La procedura utilizzata per effettuare i test descritti in questa sezione è la seguente:

- Fotografia tramite webcam ad ogni posizione del motore;
- Rielaborazione delle immagini con matlab;
- Estrazione degli angoli della scacchiera oggetto della fotografia;
- Calibrazione¹ della telecamera (tool Calibox) [5];
- Valutazione degli errori di calibrazione;
- Valutazione dei parametri estrinseci;
- Estrapolazione dell'angolo di rotazione dalle immagini;
- Confronto tra l'angolo di rotazione comandato ai motori e quello ricavato da matlab;
- Medie e valutazioni di carattere generale.

Analizziamo il test numero 18 come esempio del *modus operandi* comune a tutti i test:

Nella figura 9.1 possiamo vedere le cinque fotografie scattate con la webcam nelle cinque diverse angolazioni del motore : 0° , 4.5° , 9° , 13.5° , 18° e 22.5° . In ogni immagine è presente la scacchiera necessaria al tool di calibrazione per estrapolare i dati sulla posizione della telecamera.

Abbiamo poi analizzato ogni singola immagine per identificare i quattro angoli esterni² della scacchiera. Avendo inserito le dimensioni dei tasselli della scacchiera, matlab è stato in grado di localizzare gli angoli interni come mostrato nella figura 9.2.

¹Lo scopo fondamentale della calibrazione è il calcolo delle costanti di trasformazione di un punto della scena nelle coordinate di proiezione per ogni telecamera e nelle coordinate immagine sull'elaboratore. Sebbene si possano trarre informazioni anche da telecamere non calibrate, questa operazione è fondamentale quando è richiesta una precisa informazione spaziale come nel caso dei test proposti.

²Con *esterni* indichiamo gli angoli che permettono di identificare il maggior numero di tasselli pieni, scartiamo quindi i tasselli che fungono da bordo all'intera scacchiera per evitare eventuali imperfezioni o tasselli con una coppia di lati di dimensioni inferiori rispetto agli altri



Figura 9.1: Matrice delle immagini

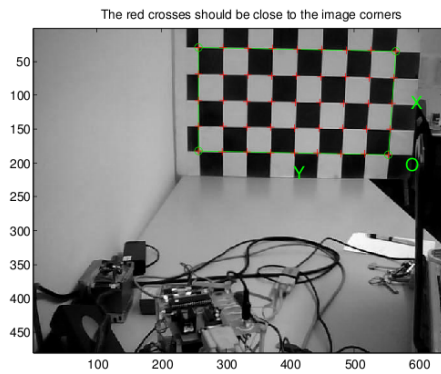


Figura 9.2: Estrazione degli angoli dalla scacchiera

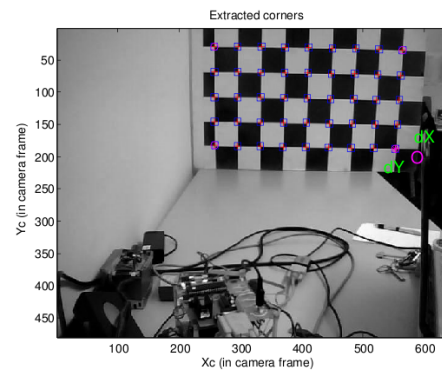


Figura 9.3: Intorni degli angoli della scacchiera

Abbiamo inserito una tolleranza di 5px nella ricerca degli angoli interni al fine di permettere al software di calibrazione di identificare un intorno in cui cercare gli angoli come si può vedere nella figura 9.3.

Dopo aver identificato tutti gli angoli delle cinque immagini è stata eseguita

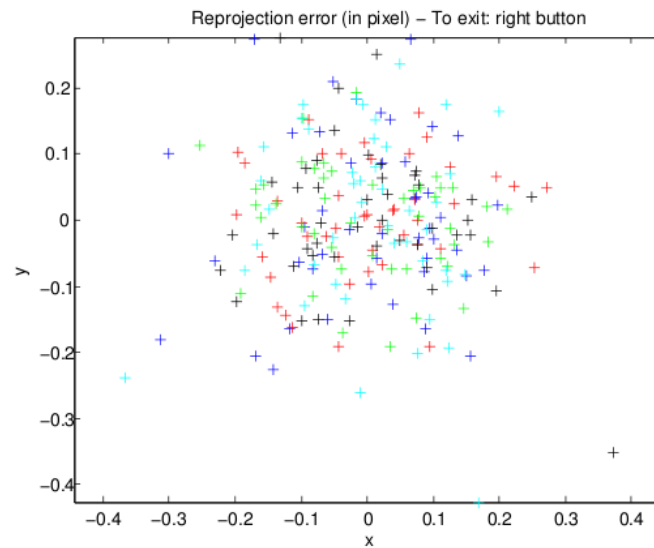


Figura 9.4: Distribuzione degli errori

la calibrazione, al termine della quale è stato possibile ottenere il grafico sulla distribuzione degli errori. Nella figura 9.4 possiamo vedere l'errore tra l'angolo reale e l'angolo calcolato da matlab, in particolare ogni errore di una stessa immagine è mostrato con lo stesso colore.

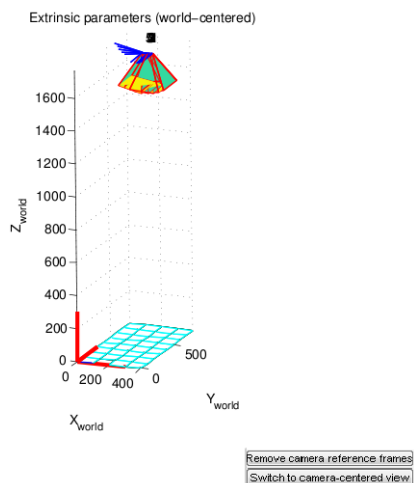


Figura 9.5: Parametri estrinseci rispetto alla scacchiera

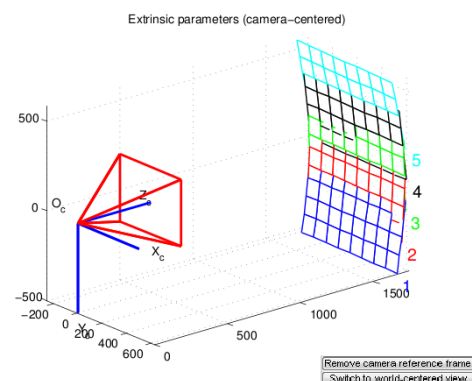


Figura 9.6: Parametri estrinseci rispetto alla camera

Abbiamo poi estrapolato i parametri delle scacchiere rispetto alle telecamere (e viceversa) come mostrato nelle immagini 9.6 e 9.5. Al termine di questa procedura abbiamo utilizzato i parametri estrinseci per calcolare l'effettiva rotazione della telecamera rispetto a due diverse immagini. Sottraendo alla misura ottenuta la dimensione del passo utilizzato è stato possibile calcolare l'errore di ogni singolo spostamento.

Dopo aver ripetuto l'operazione per i 18 test siamo stati in grado di completare la tabella 9.3 per il primo motore, la 9.4 per il secondo motore e la 9.5 per il terzo motore.

Nella tabella 9.6 infine abbiamo sintetizzato le medie dei risultati ottenuti suddividendo i risultati in base al tipo di test eseguito al fine di verificare se cambiando il tipo di pilotaggio effettuato l'errore variasse in maniera significativa. Il risultato ci dimostra che non esiste una grossa differenza tra i risultati dei diversi test dato che l'ordine di grandezza è esattamente lo stesso. L'errore medio calcolato su tutti e 18 i test è stato di 0,0547 gradi; prendendo come riferimento il passo intero da 1,8 gradi possiamo calcolare un errore percentuale medio pari a circa il 3%, valore che consideriamo pienamente soddisfacente. Abbiamo calcolato inoltre l'errore massimo medio su tutti i test, questo risulta uguale a 0.1070 gradi pari al 5.94% del passo intero. Chiamiamo E.Med., E.P.Med., E.Max. ed E.P.Max. rispettivamente l'errore medio, l'errore percentuale medio, l'errore massimo e l'errore percentuale massimo ricavato dai test.

Il risultato della tabella 9.6 non evince alcuna situazione particolarmente critica riguardo all'errore medio e massimo. Una piccola variabilità è giustificata dal numero contenuto di dati analizzati.

L'errore massimo calcolato sommando quindi errore di risoluzione ed errore sistematico è pari quindi a 0.5047° . Possiamo quindi affermare di aver rispettato le specifiche di precisione richieste.

Prova	Nr. 1	Nr. 2	Nr. 3	Nr. 4	Nr. 5	Nr. 6
Drive	normal	wave	both	normal	wave	both
Modalità	full step	full step	half step	full step	full step	half step
Passo	1,8	1,8	0,9	5,4	5,4	4,5
POSIZIONI COMANDATE (in gradi)						
1° Posizione	0	0,9	0	0	0,9	0
2° Posizione	1,8	2,7	0,9	5,4	6,3	4,5
3° Posizione	3,6	4,5	1,8	10,8	11,7	9
4° Posizione	5,4	6,3	2,7	16,2	17,1	13,5
5° Posizione	7,2	8,1	3,6	21,6	22,5	18
6° Posizione	9	9,9	4,5			22,5
7° Posizione	10,8	11,7	5,4			
8° Posizione	12,6	13,5	6,3			
9° Posizione	14,4	15,3	7,2			
10° Posizione	16,2	17,1	8,1			
11° Posizione	18	18,9	9			
12° Posizione	19,8	20,7	9,9			
13° Posizione	21,6	22,5	10,8			
PASSI RILEVATI (in gradi)						
1° Spostamento	1,7786	1,6775	0,8993	5,1676	5,2671	4,2441
2° Spostamento	1,8286	1,7844	0,8398	5,3795	5,3955	4,4313
3° Spostamento	1,8112	1,7964	0,9240	5,3909	5,4303	4,4162
4° Spostamento	1,8841	1,8029	0,8832	5,5084	5,4534	4,5037
5° Spostamento	1,7743	1,7989	0,9286			4,3824
6° Spostamento	1,8945	1,8422	0,8750			
7° Spostamento	1,8129	1,8305	0,9284			
8° Spostamento	1,8664	1,8362	0,9021			
9° Spostamento	1,8312	1,8249	0,8985			
10° Spostamento	1,8840	1,8469	0,8974			
11° Spostamento	1,8359	1,8512	0,9049			
12° Spostamento	1,8731	1,8350	0,9266			
Passo medio	1,8396	1,8106	0,9007	5,3616	5,3866	4,3955
ERRORE SUL SINGOLO PASSO (in gradi)						
1° Spostamento	0,0214	0,1225	0,0007	0,2324	0,1329	0,2559
2° Spostamento	0,0286	0,0156	0,0602	0,0205	0,0045	0,0687
3° Spostamento	0,0112	0,0036	0,0240	0,0091	0,0303	0,0838
4° Spostamento	0,0841	0,0029	0,0168	0,1084	0,0534	0,0037
5° Spostamento	0,0257	0,0011	0,0286			0,1176
6° Spostamento	0,0945	0,0422	0,0250			
7° Spostamento	0,0129	0,0305	0,0284			
8° Spostamento	0,0664	0,0362	0,0021			
9° Spostamento	0,0312	0,0249	0,0015			
10° Spostamento	0,0840	0,0469	0,0026			
11° Spostamento	0,0359	0,0512	0,0049			
12° Spostamento	0,0359	0,0512	0,0049			
Errore medio	0,0443	0,0357	0,0166	0,0926	0,0553	0,1059
Errore massimo	0,0945	0,1225	0,0602	0,2324	0,1329	0,2559

Tabella 9.3: Test eseguiti sul motore numero 1 processando le immagini scattate con la telecamera sinistra

Prova	Nr. 7	Nr. 8	Nr. 9	Nr. 10	Nr. 11	Nr. 12
Drive	normal	wave	both	normal	wave	both
Modalità	full step	full step	half step	full step	full step	half step
Passo	1,8	1,8	0,9	5,4	5,4	4,5
POSIZIONI COMANDATE (in gradi)						
1° Posizione	0	0,9	0	0	0,9	0
2° Posizione	1,8	2,7	0,9	5,4	6,3	4,5
3° Posizione	3,6	4,5	1,8	10,8	11,7	9
4° Posizione	5,4	6,3	2,7	16,2	17,1	13,5
5° Posizione	7,2	8,1	3,6	21,6	22,5	18
6° Posizione	9	9,9	4,5			22,5
7° Posizione	10,8	11,7	5,4			
8° Posizione	12,6	13,5	6,3			
9° Posizione	14,4	15,3	7,2			
10° Posizione	16,2	17,1	8,1			
11° Posizione	18	18,9	9			
12° Posizione	19,8	20,7	9,9			
13° Posizione	21,6	22,5	10,8			
PASSI RILEVATI (in gradi)						
1° Spostamento	1,7412	1,7725	0,7532	5,3898	5,4112	4,4559
2° Spostamento	1,8266	1,8242	0,8137	5,3966	5,4048	4,4965
3° Spostamento	1,8467	1,8327	0,8228	5,4021	5,4039	4,5308
4° Spostamento	1,7761	1,7547	0,8216	5,4644	5,4294	4,4679
5° Spostamento	1,8023	1,8381	0,8274			4,4977
6° Spostamento	1,8517	1,8404	0,8338			
7° Spostamento	1,8478	1,8415	0,8206			
8° Spostamento	1,7895	1,7705	0,7807			
9° Spostamento	1,8014	1,8325	0,8030			
10° Spostamento	1,8518	1,8423	0,8210			
11° Spostamento	1,8442	1,8504	0,8358			
12° Spostamento	1,8121	1,7724	0,8273			
Passo medio	1,8159	1,8144	0,8134	5,4132	5,4123	4,4898
ERRORE SUL SINGOLO PASSO (in gradi)						
1° Spostamento	0,0588	0,0275	0,1468	0,0102	0,0112	0,0441
2° Spostamento	0,0266	0,0242	0,0863	0,0034	0,0048	0,0035
3° Spostamento	0,0467	0,0327	0,0772	0,0021	0,0039	0,0308
4° Spostamento	0,0239	0,0453	0,0784	0,0644	0,0294	0,0321
5° Spostamento	0,0023	0,0381	0,0726			0,0023
6° Spostamento	0,0517	0,0404	0,0662			
7° Spostamento	0,0478	0,0415	0,0794			
8° Spostamento	0,0105	0,0295	0,1193			
9° Spostamento	0,0014	0,0325	0,0970			
10° Spostamento	0,0518	0,0423	0,0790			
11° Spostamento	0,0442	0,0504	0,0642			
12° Spostamento	0,0442	0,0504	0,0642			
Errore medio	0,0342	0,0379	0,0859	0,0200	0,0123	0,0226
Errore massimo	0,0588	0,0504	0,1468	0,0644	0,0294	0,0441

Tabella 9.4: Test eseguiti sul motore numero 2 processando le immagini scattate con la telecamera sinistra

Prova	Nr. 13	Nr. 14	Nr. 15	Nr. 16	Nr. 17	Nr. 18
Drive	normal	wave	both	normal	wave	both
Modalità	full step	full step	half step	full step	full step	half step
Passo	1,8	1,8	0,9	5,4	5,4	4,5
POSIZIONI COMANDATE (in gradi)						
1° Posizione	0	0,9	0	0	0,9	0
2° Posizione	1,8	2,7	0,9	5,4	6,3	4,5
3° Posizione	3,6	4,5	1,8	10,8	11,7	9
4° Posizione	5,4	6,3	2,7	16,2	17,1	13,5
5° Posizione	7,2	8,1	3,6	21,6	22,5	18
6° Posizione	9	9,9	4,5			22,5
7° Posizione	10,8	11,7	5,4			
8° Posizione	12,6	13,5	6,3			
9° Posizione	14,4	15,3	7,2			
10° Posizione	16,2	17,1	8,1			
11° Posizione	18	18,9	9			
12° Posizione	19,8	20,7	9,9			
13° Posizione	21,6	22,5	10,8			
PASSI RILEVATI (in gradi)						
1° Spostamento	1,7899	1,7591	0,8356	5,4401	5,4392	4,5234
2° Spostamento	1,8321	1,8330	0,8535	5,4951	5,4878	4,6332
3° Spostamento	1,8529	1,8605	0,8235	5,5378	5,5250	4,5654
4° Spostamento	1,8658	1,8527	0,8389	5,5079	5,5387	4,5659
5° Spostamento	1,8589	1,8125	0,8721			4,6336
6° Spostamento	1,8332	1,8397	0,8779			
7° Spostamento	1,8708	1,8706	0,8662			
8° Spostamento	1,8603	1,8571	0,8476			
9° Spostamento	1,8722	1,8200	0,8917			
10° Spostamento	1,8361	1,8364	0,8511			
11° Spostamento	1,8654	1,8750	0,8506			
12° Spostamento	1,8666	1,8521	0,8587			
Passo medio	1,8503	1,8391	0,8556	5,4952	5,4977	4,5843
ERRORE SUL SINGOLO PASSO (in gradi)						
1° Spostamento	0,0101	0,0409	0,0644	0,0401	0,0392	0,0234
2° Spostamento	0,0321	0,0330	0,0465	0,0951	0,0878	0,1332
3° Spostamento	0,0529	0,0605	0,0765	0,1378	0,1250	0,0654
4° Spostamento	0,0658	0,0527	0,0611	0,1079	0,1387	0,0659
5° Spostamento	0,0589	0,0125	0,0279			0,1336
6° Spostamento	0,0332	0,0397	0,0221			
7° Spostamento	0,0708	0,0706	0,0338			
8° Spostamento	0,0603	0,0571	0,0524			
9° Spostamento	0,0722	0,0200	0,0083			
10° Spostamento	0,0361	0,0364	0,0489			
11° Spostamento	0,0654	0,0750	0,0494			
12° Spostamento	0,0654	0,0750	0,0494			
Errore medio	0,0519	0,0478	0,0451	0,0952	0,0977	0,0843
Errore massimo	0,0722	0,0750	0,0765	0,1378	0,1387	0,1336

Tabella 9.5: Test eseguiti sul motore numero 3 processando le immagini scattate con la telecamera destra

Descrizione	E.Med.	E.P.Med.	E.Max.	E.P.Max.
Totale	0,0547	3,04%	0,1070	5,94%
Motore Nr.1	0,0584	3,25%	0,1497	8,32%
Motore Nr.2	0,0355	1,97%	0,0657	3,65%
Motore Nr.3	0,0703	3,91%	0,1056	5,87%
Normal Drive	0,0564	3,13%	0,1100	6,11%
Wave Drive	0,0478	2,65%	0,0915	5,08%
Full Step	0,0481	2,67%	0,0930	5,17%
Half Step	0,0601	3,34%	0,1195	6,64%
Passi Piccoli	0,0444	2,47%	0,0841	4,67%
Passi Medi	0,0651	3,62%	0,1299	7,22%

Tabella 9.6: Indici di sintesi degli errori medi e massimi

Capitolo 10

Conclusioni e sviluppi futuri

“La miglior cosa del futuro è che arriva un giorno alla volta.”

Abraham Lincoln (1809-1865), 16° presidente degli Stati Uniti, politico
americano ...

10.1 Conclusioni

Nella presente tesi abbiamo realizzato la testa robotica denominata MOE per il robot Maximum One. La realizzazione verteva in particolare sulla creazione di un sistema di visione bioispirato. Per questo motivo ci sono state fornite le seguenti specifiche:

Campo visivo di estensione simile a quelle umane;

Proporzioni degli occhi e della distanza tra gli occhi simili a quelle umane;

Velocità di rotazione pari a quella umana sia in caso di inseguimento lento che in caso di saccade.

Sono state fornite inoltre delle specifiche supplementari come:

Precisione elevata, con un errore totale tollerato pari a 0.5° ;

Risoluzione e frame rate delle telecamere adeguati per eventuali applicazioni quali object tracking e face detection;

Peso totale inferiore al chilogrammo per il montaggio sulla testa del robot Maximum One.

Il nostro lavoro quindi ha prodotto un sistema originale nei componenti, nella struttura finale e ha cercato, dove possibile, di rimanere entro un budget ridotto, in particolare nella scelta oculata di motori e webcam.

10.1.1 Punti di forza del sistema MOE

Abbiamo soddisfatto completamente le specifiche relative al campo visivo, alle proporzioni, alla precisione ed alla risoluzione delle telecamere. Per quanto riguarda le velocità di rotazione raggiungibili possiamo affermare di soddisfare le specifiche nella maggioranza delle situazioni a patto di concentrare il movimento sull'asse orizzontale o di eseguire saccadi di piccola ampiezza come spiegato nella sezione 9.2. Per quanto riguarda il peso totale superiamo le specifiche richieste di circa 300 grammi; non siamo riusciti a ridurre il peso totale oltre i 1.3 chilogrammi, tuttavia il peso totale della struttura così come la velocità massima raggiungibile sono specifiche facilmente migliorabili nel caso si decidesse di sostituire il motore adibito al movimento verticale con un nuovo stepper più costoso ma con prestazioni superiori ed un peso notevolmente inferiore. Merita una nota particolare la precisione del sistema estremamente elevata; il sistema MOE infatti ha un errore massimo inferiore al mezzo grado in qualsiasi posizione richiesta ed addirittura di circa 0.05° per posizioni multiple di 0.9 gradi.

10.1.2 Punti deboli del sistema MOE

Come descritto nella sezione precedente le specifiche rispetto al peso ed alla velocità di rotazione non sono attualmente soddisfatte. Nella costruzione del sistema MOE abbiamo dato particolare importanza alla precisione del sistema, per questo motivo ogni sistema meccanico che avrebbe potuto introdurre giochi è stato ridotto al massimo ed in alcuni casi scartato. Se da un punto di vista di precisione il sistema è risultato pienamente soddisfacente dal punto di vista del bilanciamento potrebbero sorgere alcuni problemi. Tra questi la testa sbilanciata, a causa del motore adibito al movimento verticale, potrebbe necessitare delle correzioni nel controllo del movimento del collo del robot Maximum One. Inoltre anche le dinamiche del movimento potrebbero non essere ideali nel caso venissero usate testa e collo contemporaneamente. Un secondo problema riscontrato è quello relativo all'aspetto scarsamente umanoide di MOE; anche in questo caso la motivazione è dettata dalla volontà di evitare giochi e quindi imprecisioni, altrimenti sarebbero state sviluppate le idee presenti nei primi prototipi. Un ulteriore problema è quello del rumore del motore verticale che durante il controllo *fischia* e potrebbe disturbare l'eventuale acquisizione audio da parte delle telecamere.

10.2 Possibili miglioramenti

Abbiamo valutato che, per migliorare il sistema, potrebbero essere effettuate le seguenti modifiche:

1. sostituire motore verticale;
2. implementare controllo in microstepping;
3. cambiare la parte meccanica con materiali più leggeri o una struttura differente.

10.2.1 Sostituzione motore verticale

Il motore verticale attuale presenta numerosi elementi di criticità che derivano da una bassa qualità manifatturiera di cui il rumore sopracitato è una chiara prova. Purtroppo il rumore non è l'unico contro di questo dispositivo. Ben più grave probabilmente è il basso valore di coppia erogata. Ciò limita la velocità massima di rotazione raggiungibile dal carico. Sicuramente sostituire il motore con uno più prestante sotto quest'ultimo aspetto aiuterebbe a centrare meglio il requisito di velocità, ad oggi non a pieno raggiunto. La sostituzione del motore verticale non comporterebbe molto lavoro per quanto riguarda l'adeguamento della parte elettronica, a patto che il nuovo motore sia un bipolare a 4 fili, e richieda una corrente massima per avvolgimento inferiore ai 2A. In questo caso basterebbe regolare il partitore di tensione del circuito chopper per controllare la corrente in ingresso agli avvolgimenti. In previsione di ciò, abbiamo fornito di un trimmer il partitore di tensione sulla scheda adibita al motore principale. Tramite un cacciavite e un tester è possibile quindi regolare la V_{sens} in ingresso all'integrato L297 e alimentare il nuovo motore con la corrente desiderata. I valori di V_{sens} e le rispettive correnti sono illustrate nell'articolo [27]. Per quanto riguarda la parte meccanica, la sostituzione del motore comporterebbe solo ulteriori sforzi per lo spostamento delle attuali placche di alluminio che lo reggono. Un possibile esempio di motore adatto è il *Size 16 hybrid stepper motor* di cui alleghiamo il datasheet e presente anch'esso su RS.

10.2.2 Controllo in microstepping

Nella sezione 9 abbiamo dimensionato gli errori di precisione del sistema MOE. Si evince che il maggior contributo a tale errore deriva dalla risoluzione (passi da $0,9^\circ$). Se riuscissimo a aumentare la risoluzione potremmo ridurre sensibilmente l'errore di precisione. Ciò è ottenibile realizzando il controllo in

microstepping dei motori passo-passo. Tale controllo consiste nell'aumentare gli stati di pilotaggio aggiungendo per ogni stato attuale degli stati intermedi ottenibili variando la corrente su ogni singolo avvolgimento. Implementare tale controllo è complesso e richiede di modificare radicalmente l'apparato elettronico. Bisognerebbe spostare a livello del PIC la logica applicativa della gestione degli stati di pilotaggio (attualmente svolta dall'integrato L297), cablare due input analogici per ogni motore sulle Vsens e realizzare via software il circuito chopper. Maggiori informazioni si possono trovare nel riferimento [28]

10.2.3 Miglioramento della parte meccanica

La parte meccanica sviluppata artigianalmente in laboratorio potrebbe essere migliorata mediante due diversi approcci. Il primo approccio riguarda una diversa struttura e disposizione dei motori, in particolare riprendendo le idee esposte nei primi prototipi analizzati nella sezione 5. Al fine di migliorare il bilanciamento della struttura per un più comodo posizionamento sul collo esistente infatti, potrebbero essere utilizzate delle pulegge o dei cavi inestensibili per permettere alla parte mobile contenente le telecamere di ruotare. Questa soluzione potrebbe essere utile anche per un eventuale controllo combinato collo-testa ma porterebbe inevitabilmente a un deterioramento della precisione del sistema.

Il secondo approccio invece riguarda la ricreazione della struttura in materiale plastico monoblocco modellato ad HOC sulla struttura esistente. Tale soluzione abbasserebbe notevolmente il peso della struttura che attualmente è formata in buona parte da placche in alluminio e numerose viti con i corrispondenti bulloni. Non è da sottovalutare tuttavia l'attuale funzione di dissipazione del calore dei motori che attualmente tali placche in alluminio svolgono.

10.3 Sviluppi futuri e varianti

Esistono inoltre possibili sviluppi futuri e varianti del tema in esame al fine di completare l'analisi effettuata. Tali sviluppi riguardano la sostituzione dei motori passo passo con una muscolatura artificiale, l'abbinamento del movimento del collo e dello sviluppo della computer vision.

10.3.1 Muscoli di McKibben

L'utilizzo di muscoli di McKibben rientra nell'idea di un robot totalmente bioispirato. Questa possibilità è stata scartata in fase progettuale per i

notevoli problemi relativi al controllo. Un sistema dotato di motori stepper risulta più preciso nei movimenti e più semplice nella costruzione del sistema meccanico. I motori stepper inoltre ci permettono quando non sono in movimento di mantenere fermamente la posizione, grazie alla resistenza creata dal passaggio di corrente negli avvolgimenti. Nel caso venissero usati muscoli di McKibben bisognerebbe compensare l'effetto della gravità sulla struttura meccanica. Come mostrato nella sezione 2.2.1 delle telecamere attuate da muscoli artificiali sono un'opzione possibile tuttavia non esistono delle analisi sulle prestazioni dei muscoli artificiali e per questo motivo riteniamo difficile raggiungere le velocità necessarie ad eseguire movimenti saccadici efficienti come quelli attuali. Infine il calcolatore ove attualmente è collegato il robot Maximum One presenta una sola uscita seriale insufficiente per il controllo di tutti i muscoli contemporaneamente, nell'ottica di utilizzare una muscolatura di McKibben bisogna preventivare la sostituzione di tale scheda con una dotata di un numero maggiore di uscite.

10.3.2 Integrare il movimento del collo

In un'ottica di progetto biospirato è di particolare interesse l'interazione del collo con la testa oggetto della nostra lavoro. Per fare questo sono numerose le considerazioni da fare:

- il collo riesce effettivamente a mantenere il sistema fermo?
- il collo riesce a muovere la testa correttamente?
- il collo riesce ad eseguire rotazioni a destra e sinistra?
- è possibile controllare collo e testa contemporaneamente? Se sì, come?
- è possibile incrementare le velocità massime raggiungibili dal sistema abbinando collo e testa?
- quali errori di posizione otteniamo dal sistema combinato collo-testa?

Una possibile evoluzione futura potrebbe essere la verifica di queste considerazioni sopra esposte.

10.3.3 Computer vision

Nell'ambito della *computer vision* un progetto che è già in fase di studio è la realizzazione di un software in matlab per la gestione ad alto livello

dei movimenti che tenga conto del disaccoppiamento delle due camere ¹ e dell'errore di posizione. Dato che una prima analisi dell'errore è stata fatta, si potrebbe approfondire tale analisi con l'ausilio del software per matlab e verificare eventuali dipendenze dalla velocità di spostamento e dalla posizione di partenza del movimento, non più solo dalla quantità di spostamento stesso. Il software poi fungerebbe da layer intermedio di raccordo tra il software di MOE, descritto nel seguente elaborato ed altri progetti quali object tracking o convergenza dei due occhi per la messa a fuoco a diverse distanze. Questi progetti invocherebbero il software di raccordo che a sua volta comunicherebbe con il software utilizzato attualmente per MOE e che gestisce i movimenti delle telecamere in maniera diretta. Sarà necessario quindi una ridefinizione di un sistema di riferimento fisso (che può essere ad esempio sul collo) a cui legare i sistemi di riferimento delle due camere e bisognerà verificare la calibrazione delle camere per gestire l'errore di posizione. Per fare questo verranno usati alcuni algoritmi già esistenti.

Video tracking

Il video tracking è il processo di individuazione di uno o più oggetti in movimento per mezzo di una telecamera. L'object tracking ha una varietà di possibili applicazioni come l'interazione uomo-macchina, sicurezza, sorveglianza, comunicazione e compressione video, controllo del traffico, video editing, etc. L'obiettivo del video tracking è quello di associare oggetti presenti in fotogrammi consecutivi, nel caso di movimenti lenti l'oggetto sarà in posizioni simili in fotogrammi consecutivi, diversamente nel caso di movimenti molto veloci la distanza tra l'oggetto nei due fotogrammi potrebbe essere elevata rendendone difficile il riconoscimento. Al fine di analizzare sequenzialmente i frame dei video esistono numerosi algoritmi ciascuno caratterizzato da pregi e difetti. Per questo motivo la scelta dell'algoritmo da usare è fortemente legata all'applicazione che si intende perseguire. I due maggiori settori del video tracking sono :

Target Representation and Localization attraverso cui identificare gli oggetti in movimento. Gli algoritmi più comuni in questo senso sono :

Blob tracking che consiste nel ricondurre l'oggetto ad una regione di interesse per l'immagine denominata *blob* e ricavarne il centro di massa per eseguire inseguimento.

¹i sistemi di riferimento non hanno origine sullo stesso asse di rotazione, per la configurazione del meccanismo stesso

Kernel-based tracking che consiste nella localizzazione iterativa basata sul riconoscimento di similitudini nell'immagine per un tempo variabile.

Contour tracking che consiste nel riconoscimento dei contorni

Filtering and Data Association attraverso cui identificare oggetti complessi. Gli algoritmi (filtri) più utilizzati sono :

Kalman filter un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico a partire da una serie di misure soggette a rumore.

Particle filter o metodo di Monte Carlo è usato per trarre stime attraverso simulazioni e si basa su di un algoritmo che genera una serie di numeri tra loro non correlati, che seguono la distribuzione di probabilità che si suppone abbia il fenomeno da indagare.

Una possibile applicazione del video tracking alla presente tesi è il riconoscimento di oggetti in movimento ed il loro inseguimento per mantenerli al centro dell'immagine delle telecamere in questo senso l'object tracking a mezzo di un algoritmo di blob tracking potrebbe essere la scelta adatta al nostro sistema.

10.3.4 Nuovi test

Nella capitolo 9 abbiamo verificato il funzionamento delle velocità, delle posizioni e soprattutto l'errore di posizione in funzione dello spostamento, un possibile sviluppo futuro potrebbe consistere in una nuova serie di test atti a verificare se l'errore non dipenda solo dallo spostamento ma anche dalla velocità con cui tale spostamento avviene e dal punto dal quale ha origine. Un secondo test possibile è la calibratura della struttura meccanica, restano da capire infatti le rototraslazioni che entrano in gioco nella completa caratterizzazione della struttura meccanica.

Bibliografia

- [1] *MIL-STD-1472F, Military Standard, Human Engineering, Design Criteria For Military Systems, Equipment, And Facilities*. 1999.
- [2] Bionics. <http://en.wikipedia.org/wiki/Bionics>, 2010.
- [3] A. G. Pipe Chris Melhuish Paul Dean Alexander Lenz, Sean R. Anderson and John Porrill. Cerebellar-inspired adaptive control of a robot eye actuated by pneumatic artificial muscles. 2009.
- [4] P.P. Battaglini. Motilità estrinseca dell'occhio. <http://fc.units.it/ppb/visione/blocco6.html>.
- [5] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2008.
- [6] Dr. E. Fernandez Dr. H. Kolb B.W.Jones, Dr. R. Nelson. Webvision: Simple anatomy of the retina. <http://education.vetmed.vt.edu/Curriculum/VM8054/EYE/RETINA.HTM>, 2005.
- [7] Dr. Thomas Caceci. The retinal tunic. <http://webvision.med.utah.edu/sretina.html>, 2001.
- [8] ArchLinux Community. Touchpad synaptics. http://wiki.archlinux.org/index.php/Touchpad_Synaptics.
- [9] P. Dario. *Biorobotica a Biomeccatronica, in Treccani il libro dell'anno, Istituto dell'Enciclopedia Italiana*. 2006.
- [10] S.I.M.G. Società Italiana di Medicina Generale. *Nuova enciclopedia della salute*. Gruppo editoriale l'Espresso Spa, 2002.
- [11] Rafael C. Gonzalez e Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2002.
- [12] Adriano Gaspani. Il potere risolutivo ad occhio nudo. <http://www.brera.inaf.it/utenti/gaspani/pr.htm>.

-
- [13] M.I.T. Gorge A. Bakey. *Autonomous Robots From Biological Inspiration to Implementation and Control*. 2005.
 - [14] Kristof Goris, Jelle Saldien, Bram Vanderborght, and Dirk Lefeber. The huggable robot probot: design of the robotic head. 2003.
 - [15] Kristof Goris, Jelle Saldien, Bram Vanderborght, and Dirk Lefeber. The huggable robot probot web site. <http://probo.vub.ac.be/>, 2003.
 - [16] J.E Hoffman and Subramaniam. The role of visual attention in saccadic eye movement. 1995.
 - [17] J.M Jindlay and Gilchrist. Active vision. 2003.
 - [18] Microchip. Pic16f87x data sheet. 2001.
 - [19] M.M.Gavrilovic and M.R.Maric. Positional servo-mechanism activated by artificial muscules. 1969.
 - [20] A.Bugra Koku Orhan Olcucuglu and E.Ilhan Konukseven. i-rok: A human like robotic head. 2008.
 - [21] C. Cobelli P. Dario, E. Biondi. *Robotica Biomedica, Storia della Bioingegneria*. Patron Editore, 2001.
 - [22] Miguel Praca Jose Santos-Victor Alexandre Bernardino Giorgio Metta Francesco Becchi Roque Saltaren Ricardo Beira, Manuel Lopes. Design of the robot-cub (icub) head. 2006.
 - [23] Miguel Praca Jose Santos-Victor Alexandre Bernardino Giorgio Metta Francesco Becchi Roque Saltaren Ricardo Beira, Manuel Lopes. icub. <http://www.icub.org/>, 2006.
 - [24] RS. Hybrid stepper motor datasheet. 1997.
 - [25] H. F. Schulte. The characteristics of the mckibben artificial muscle. 1961.
 - [26] Jeroen Smeets, , Jeroen B. J. Smeets, and Ignace T. C. Hooge. Nature of variability in saccades. 2003.
 - [27] STMicroelectronics. L297 datasheet. 2001.
 - [28] STMicroelectronics. L298 datasheet. 2001.
 - [29] Maria Teresa Tuccio. Meccanismo della visione: dall' occhio al cervello. <http://www.amedeolucente.it/meccanismovisione.html>, 2010.

- [30] Andy White. Headroid1 : a face tracking robot head. <http://ianozsvald.com/2010/05/21/headroid1-a-face-tracking-robot-head/>.

Appendice A

Il manuale utente

Il manuale utente si compone di due parti, la prima relativa alla messa in funzione della struttura meccanica compresi fili e attacchi; la seconda riguarda la documentazione della libreria libmoe, per la realizzazione software di controllo dell'apparato MOE.

A.1 Connessione dei cavi

Descriviamo ora i componenti di MOE:

1. Testa vera e propria, struttura dotata di 3 motori, 2 telecamere, 6 cavi schermati e 2 cavi usb per le webcam
2. Millefori multipiano relativa alla parte elettronica presenta 6 ingressi per i sensori (di cui 3 utilizzati), 3 uscite per i motori, 1 ingresso seriale maschio , 1 ingresso per l'alimentazione e 2 uscite per il tester ed un trasformatore collegato alla struttura.
3. 1 cavo seriale femmina-maschio
4. 1 adattatore seriale femmina usb
5. 1 alimentatore
6. 1 tester¹

Potete vedere nella figura A.1 la millefori multipiano prima del collegamento dei cavi, potete notare in basso a destra le tre prese dei motori², sul piano

¹facoltativo

²rispettivamente denominate: MOT.1, MOT.2, MOT.3

superiore da destra verso sinistra troviamo l'ingresso seriale maschio, il foro per l'inserimento dell'alimentazione ed i sei ingressi per i sensori: Nella figura

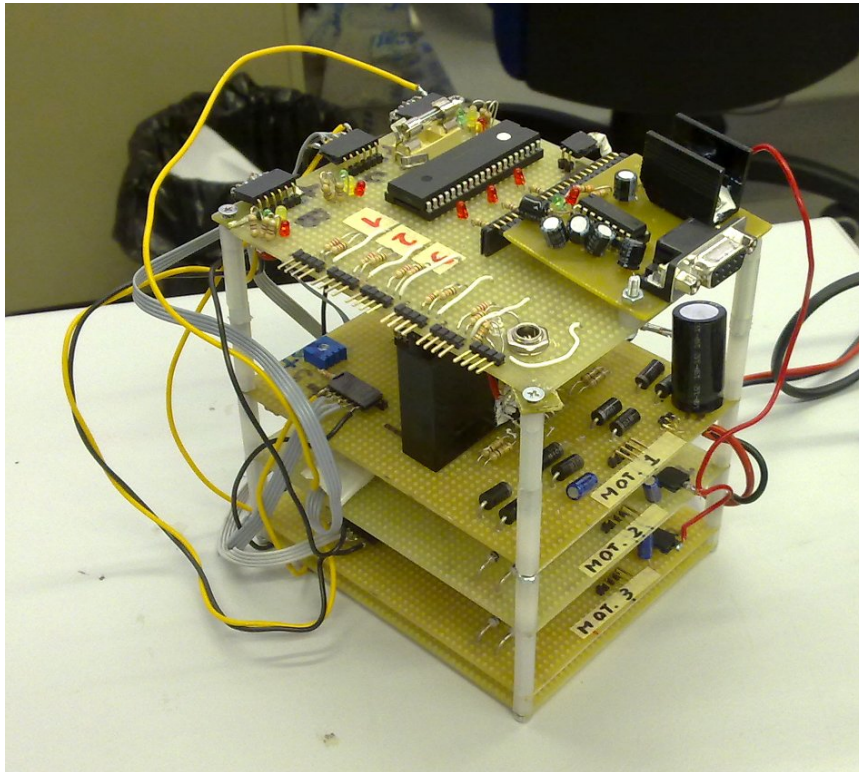
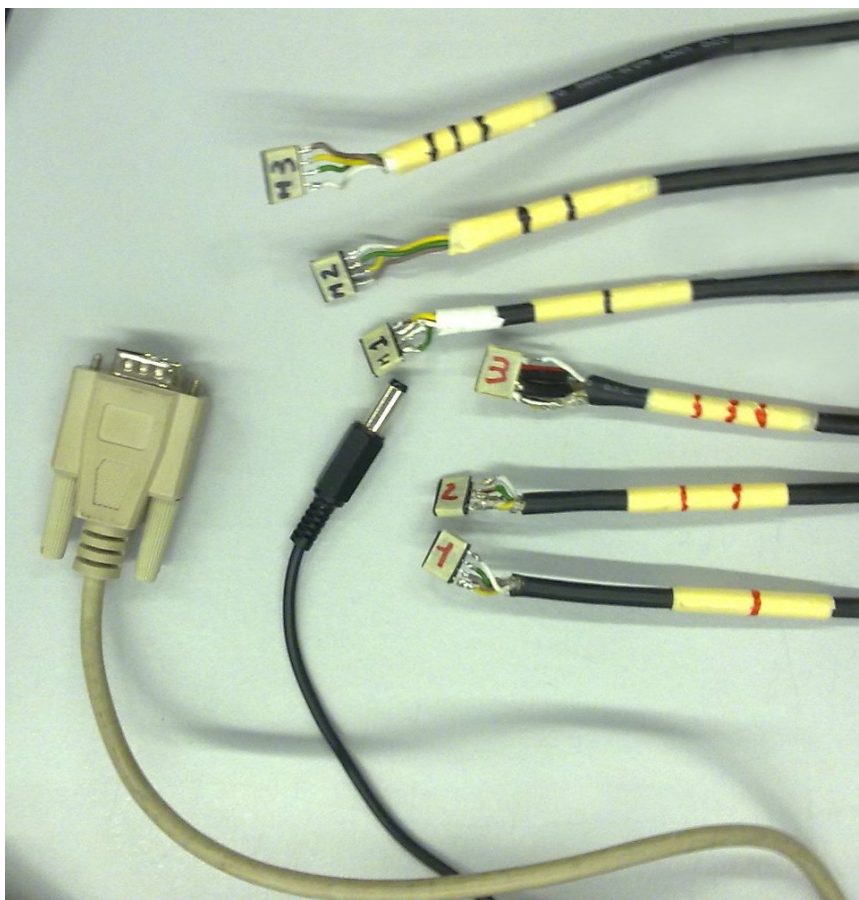


Figura A.1: Nessun cavo collegato

A.2 possiamo vedere i cavi da collegare al sistema, in senso orario troviamo l'attacco del terzo, secondo e primo motore, l'attacco del terzo, secondo e primo sensore, il cavo dell'alimentazione ed il cavo seriale. Il cavo seriale termina nell'altra estremità con il convertitore seriale usb da collegare al computer. Per quanto riguarda l'alimentazione è necessario collegare alla corrente sia il trasformatore che l'alimentatore. Mostriamo in particolare il collegamento dei motori nella figura A.3 e dei sensori nella figura A.4: La configurazione finale quindi è quella mostrata nella figura A.5: E' possibile inoltre inserire il tester come mostrato nella figura A.6 : .

*Figura A.2: Cavi*

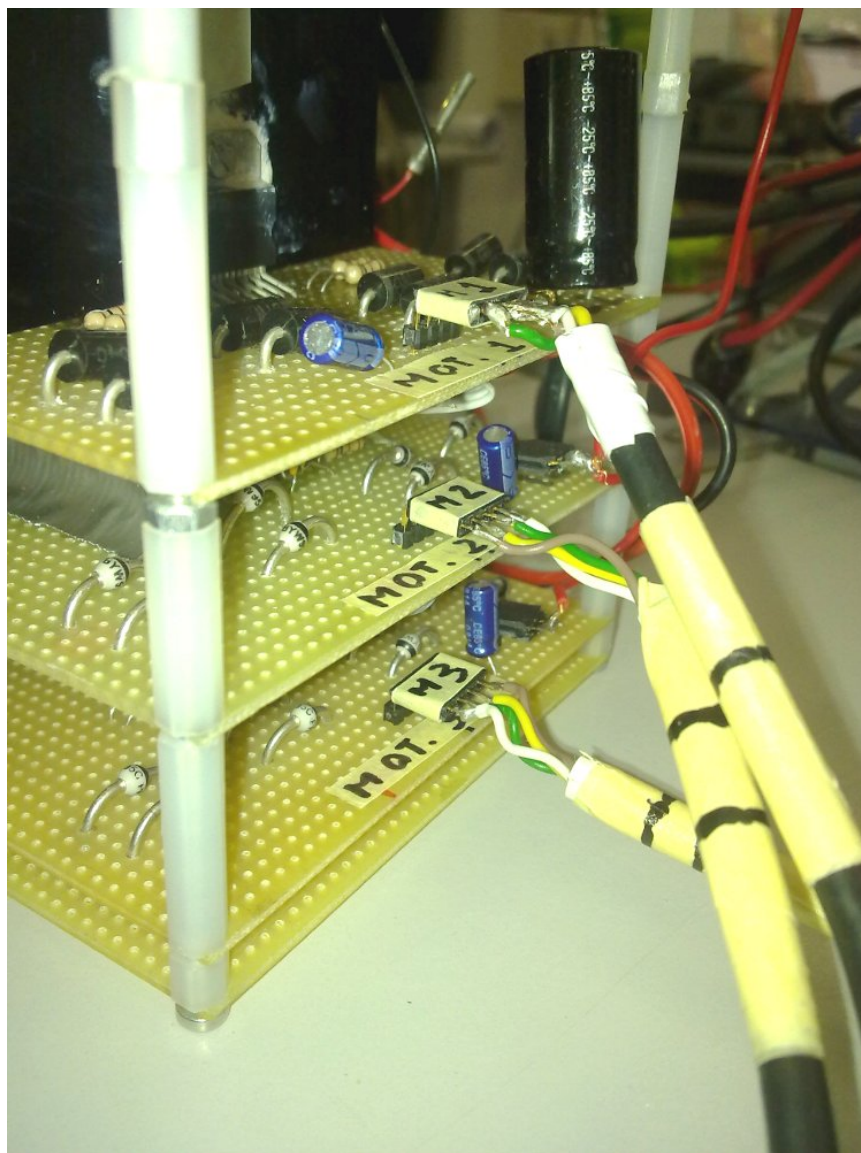


Figura A.3: Collegamento motori

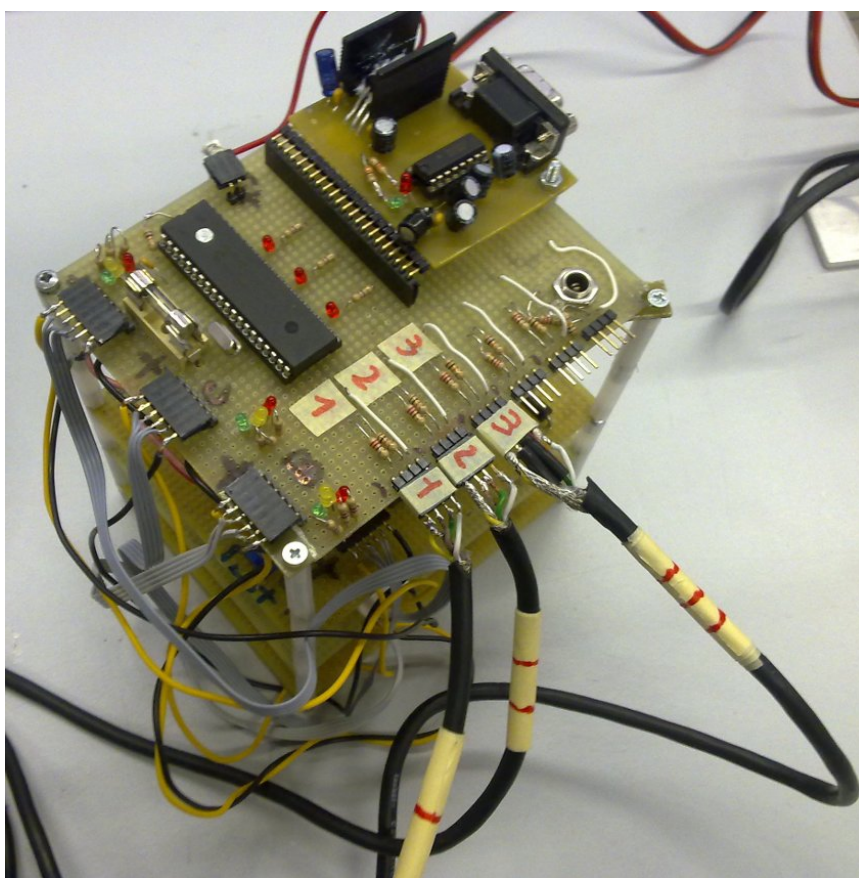


Figura A.4: Collegamento sensori

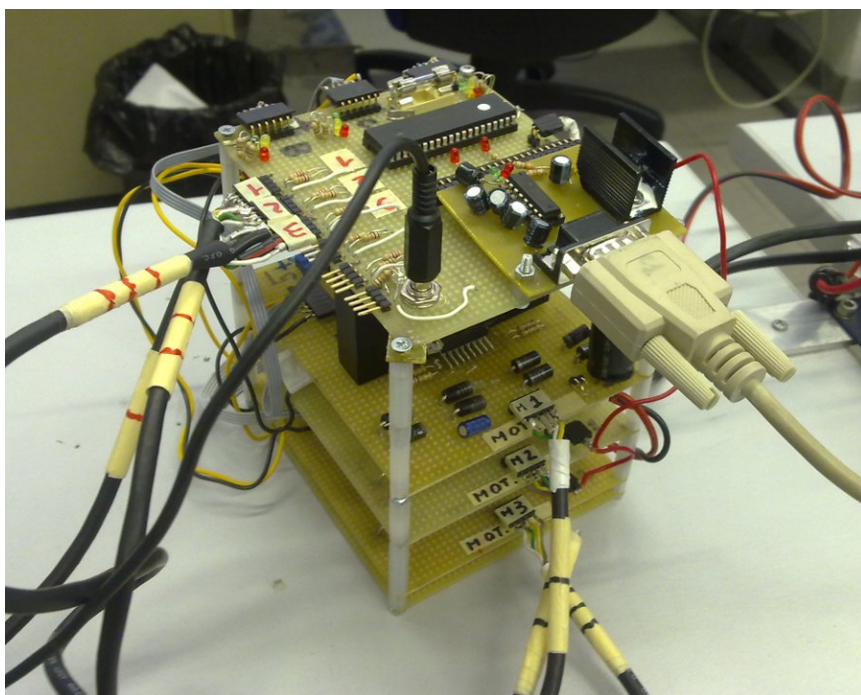


Figura A.5: Collegamento completato

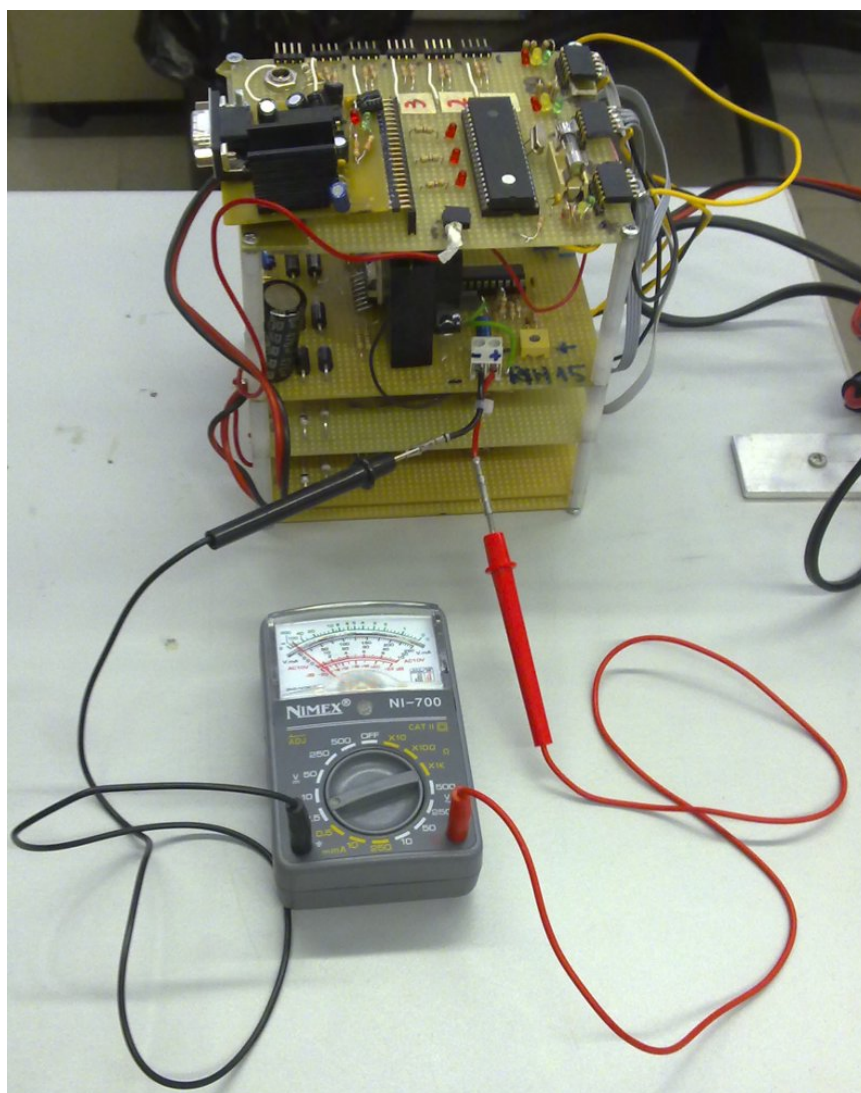


Figura A.6: Collegamento tester

A.2 Documentazione libmoe

La documentazione tratta nel dettaglio le funzioni disponibili della libreria libmoe. Per un'approfondimento sulla libreria libmoe si rimanda alla sezione 8.4.

A.2.1 parse_csv.h

```
1 static FILE * csv_x_ptr = 0;
2 static FILE * csv_y_ptr = 0;
3
4 int init_csv(char * path_x, char * path_y);
```

La funzione `init_csv` inizializza i descrittori ai file `csv_x_ptr` e `csv_y_ptr` delle tabelle di velocità. Input: `path_x` è il path alla tabella della velocità motori x1 e x2 in formato csv. `Path_y` è il path alla tabella della velocità motore y in formato csv. Output: 1 in caso di successo, 0 se almeno uno dei due csv non è stato aperto.

```
1 struct speedtable_csv
2 {
3     int speed;
4     int driving_value;
5     int enable;
6 };
7 struct speedtable_csv speedtable_x_csv[255];
8 struct speedtable_csv speedtable_y_csv[255];
9
10 void parse_csv();
```

La funzione `parse_csv` carica nelle struct `speedtable_csv[]` i dati contenuti nelle tabelle di velocità. Il campo `speed` indica la velocità dimensionata in grad/sec. Il campo `driving_value` è il corrispettivo valore utilizzato dal firmware per il controllo in velocità. Il campo `enable` indica se la coppia `speed-driving_value` viene utilizzata oppure scartata. `Parse_csv` va invocata dopo `init_csv`.

```
1 int close_csv();
```

La funzione `close_csv` chiude i descrittori ai file delle tabelle di velocità. Output: 1 in caso di successo, 0 in caso di errore

A.2.2 utils.h

```
1 unsigned char convertToStep(double deg);
```

La funzione `convertToStep` converte un valore da gradi nel formato usato dal firmware per il controllo in posizione. Input: valore della posizione da convertire in gradi, in doppia precisione. Output: valore della posizione convertita nel formato processabile dal firmware

```
1 double convertToDeg(unsigned char step);
```

La funzione `convertToDeg` converte un valore da formato usato dal firmware per il controllo in posizione a gradi. Input: valore della posizione da convertire nel formato usato dal firmware, in unsigned char. Output: valore della posizione convertita in gradi

```
1 int init_speed_table(char * path_speed_table_x, char *  
    path_speed_table_y);
```

La funzione consiste nella chiamata in serie di `init_csv` e `parse_csv`. Input: `path_speed_table_x` è il path alla tabella della velocità motori x1 e x2 in formato csv. `Path_speed_table_y` è il path alla tabella della velocità motore y in formato csv. Output: 1 in caso di successo, 0 se almeno uno dei due csv non è stato aperto.

```
1 unsigned char get_speed_from_table_x(double spd);
```

La funzione `get_speed_from_table_x` permette di selezionare la velocità nel formato processabile dal firmware del motore orizzontale che minimizza l'errore di rappresentazione (ovvero la quantità $\text{abs}(\text{spd} - \text{spd\_tabella\_di\_velocità})$) sul riferimento passato come input, dimensionato in grad/sec. Input: la velocità di riferimento, in gradi al secondo Output: la velocità rappresentabile dal firmware

```
1 unsigned char get_speed_from_table_y(double spd);
```

La funzione `get_speed_from_table_y` permette di selezionare la velocità nel formato processabile dal firmware del motore verticale che minimizza l'errore di rappresentazione (ovvero la quantità $\text{abs}(\text{spd} - \text{spd\_tabella\_di\_velocità})$) sul riferimento passato come input, dimensionato in grad/sec. Input: la velocità di riferimento, in gradi al secondo Output: la velocità rappresentabile dal firmware

```
1 int get_real_speed_from_table_x(double spd);
```

La funzione `get_real_speed_from_table_x` ritorna la velocità in gradi al secondo del motore orizzontale che minimizza l'errore di rappresentazione (ovvero la quantità `abs( spd - spd_tabella_velocità )`) sul riferimento passato come input, dimensionato in grad/sec. Input: la velocità di riferimento, in gradi al secondo Output: la velocità rappresentabile in gradi/sec

```
1 int get_real_speed_from_table_y(double spd);
```

La funzione `get_real_speed_from_table_y` ritorna la velocità in gradi al secondo del motore verticale che minimizza l'errore di rappresentazione (ovvero la quantità `abs( spd - spd_tabella_velocità )`) sul riferimento passato come input, dimensionato in grad/sec. Input: la velocità di riferimento, in gradi al secondo Output: la velocità rappresentabile in gradi/sec

A.2.3 command.h

```
1 static int MOE_fd = 0;
2
3 int MOE_open(char * path_speed_table_x ,char *
    path_speed_table_y );
```

La funzione `MOE_open` inizializza le tabelle delle velocità e il device del convertitore usb-seriale con cui il PC viene collegato fisicamente al PIC. La funzione itera ogni device con nome `ttyusbX` dove X è un intero compreso tra 0 e 9. Se un dev viene trovato, esso è inizializzato come una seriale con velocità 19200 baud, 8bit per simbolo, senza parità, senza controllo di flusso. La funzione prova quindi a inviare un comando di get, e se riceve risposta entro un timeout di 5 secondi accetta il device come valido. Input: i path delle tabelle di velocità Output: 1 in caso di successo, 0 in caso di insuccesso. Un insuccesso può dipendere sia dal path errato di una delle due tabelle di velocità, sia dal device del convertitore usb-seriale non trovato.

```
1 float * MOE_get();
```

La funzione `MOE_get` deve essere chiamata dopo che la `MOE_open` è stata eseguita con successo. Essa invia al firmware il comando di get. Output: un array di 3 valori in doppia precisione rappresentanti le posizioni in gradi delle webcam y, x1 e x2 rispettivamente.

```
1 void MOE_reset(float degOnSec);
```

La funzione `MOE_reset` deve essere chiamata dopo che la `MOE_open` è stata eseguita con successo. Essa invia al firmware il comando di reset. Tale operazione riporta le webcam alla posizione iniziale (0,0,0). Tale comando è

bloccante, eventuali altri comandi inviati durante un reset verranno accodati e eseguiti non appena terminato. Input: riferimento di velocità con cui pilotare il movimento in gradi al sec

```
1 void MOE_set(float pos_y, float pos_x1, float pos_x2, float
    degOnSec);
```

La funzione MOE_set deve essere chiamata dopo che la MOE_open è stata eseguita con successo. Essa invia al firmware il comando di set. Tale operazione porterà le webcam nella posizione desiderata, inviata come parametro. Il parametro di velocità indica la velocità del movimento composto delle due componenti orizzontale e verticale, per ogni webcam. Il calcolo della velocità viene effettuato tramite il seguente algoritmo - in pseudocodice:

```
1 //riferimento webcam y
2 double pos_y;
3
4 //riferimento webcam x1
5 double pos_x1;
6
7 //riferimento webcam x2
8 double pos_x2;
9
10 //riferimento di velocità
11 double degOnSec;
12
13 // richiedo la posizione attuale e la salvo nell'oggetto
    msg_rcvd
14 msg_rcvd = MOE_get()
15
16 //offset movimento verticale in gradi
17 delta_y = fabs(pos_y - convertToDeg( msg_rcvd.pos_y));
18
19 //offset movimento orizzontale su motore 1 in gradi
20 delta_x1 = fabs(pos_x1 - convertToDeg( msg_rcvd.pos_x1));
21
22 //offset movimento orizzontale su motore 2 in gradi
23 delta_x2 = fabs(pos_x2 - convertToDeg( msg_rcvd.pos_x2));
24
25 //angolo formato dall'asse x1 e dal vettore movimento nel
    piano cartesiano x1,y
26 tetal = atan2(delta_y, delta_x1);
27
28 //angolo formato dall'asse x2 e dal vettore movimento nel
    piano cartesiano x1,y
```

```
29 teta2 = atan2(delta_y, delta_x2);
30
31 //angolo medio
32 tetaAvg = (teta1 + teta2) / 2;
33
34 //velocità teorica motore y
35 spd_y = degOnSec * sin(tetaAvg);
36
37 //velocità teorica motore x
38 spd_x = degOnSec * cos(tetaAvg);
39
40 //velocità reale rappresentabile dal firmware per motore y
41 speed_y_real = (double) get_real_speed_from_table_y(spd_y);
42
43 //velocità reale rappresentabile dal firmware per motore x
44 speed_x_real = (double) get_real_speed_from_table_x(spd_x);
```

```
1 void MOE_close();
```

La funzione MOE_close serve per chiudere il descrittore al device del convertitore usb-seriale.

Appendice B

Datasheet

Datasheet di riferimento:

- Hybrid stepper motors
- High Torque Stepper
- Dual Full-Bridge Driver
- Stepper Motor Controllers
- Size 16 hybrid stepper motor ¹

¹possibilme motore futuro, maggiore coppia, minore peso e dimensioni

RS

Data Sheet

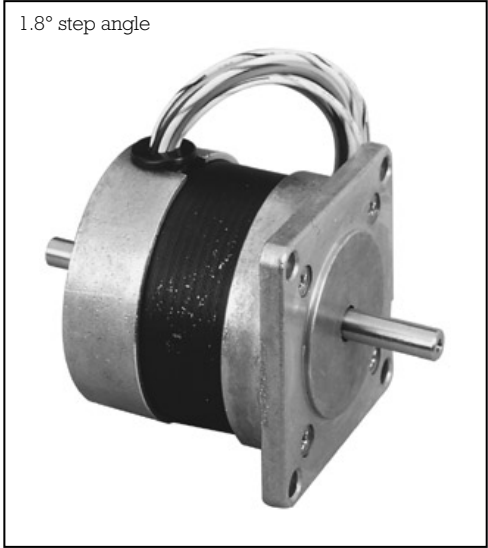
Hybrid stepper motors

Size	Rear shaft	No. of wires	RS stock no.
17	No	6	440-420
	Yes		440-436
	No		191-8299
	No		191-8306
23	No	8	440-442
	Yes	8	440-458
	No	6	191-8328
	No	6	191-8334
	No	6	191-8340
	No	6	191-8356
	No	6	191-8362
	No	8	191-8378
34	Yes	8	440-464
	No	8	440-470

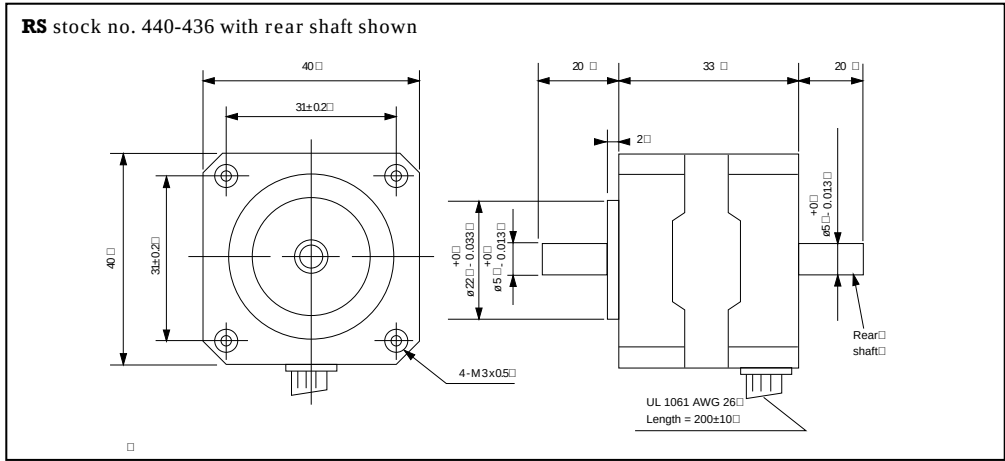
These 4 phase hybrid stepper motors are capable of delivering much higher working torques and stepping rates than permanent magnet (7.5° and 15°) types. Whilst at the same time maintaining a high detent torque even when not energised. This feature is particularly important for positional integrity. Many of the motors are directly compatible with the RS stepper motor drive boards (RS stock nos. 332-098, 342-051 and 440-240).

Size 34 motors and a number of size 23 motors are supplied in 8-lead configuration which allows the maximum flexibility when connecting to the drive boards.

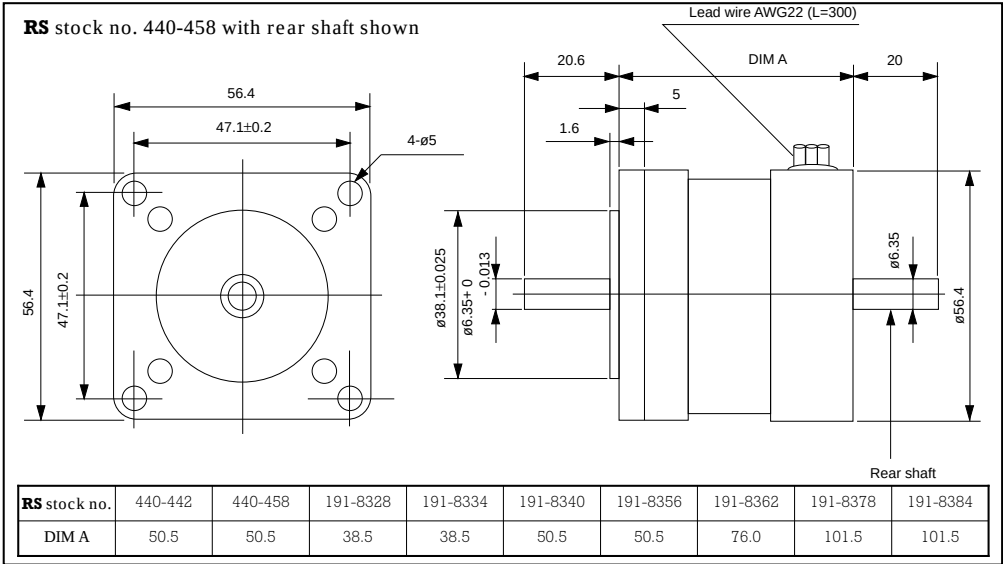
Rear extension shafts are provided on three of the motors to enable connection of other drive requirements and feedback devices.



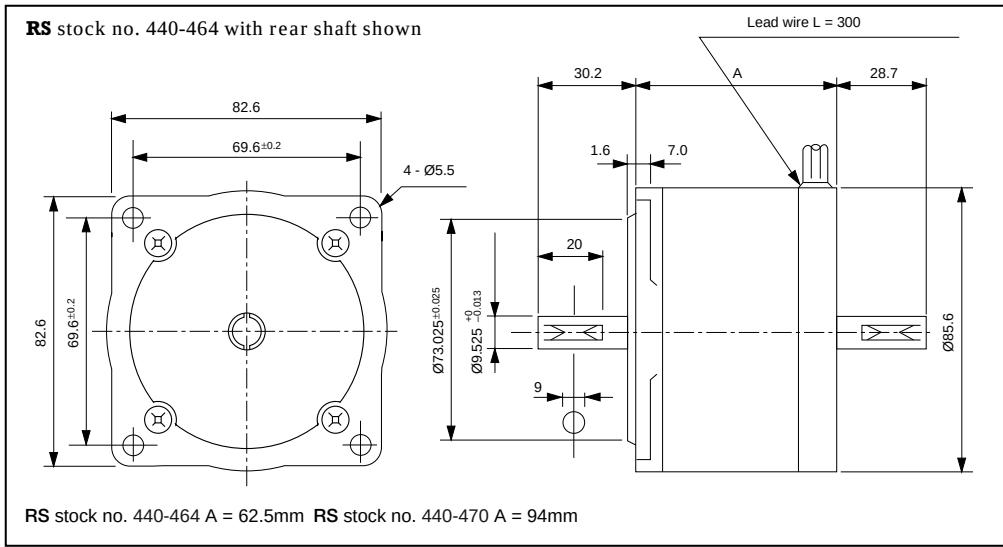
Size 17



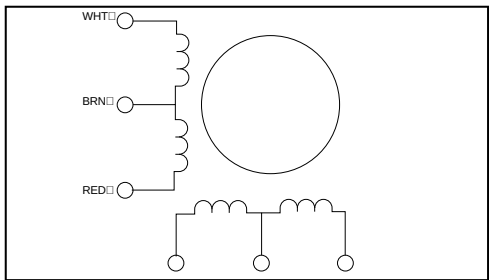
Size 23



Size 34

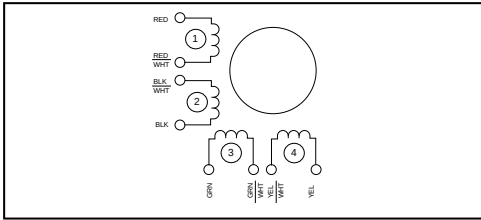


6 Wire configuration



Exciting sequence and direction of rotation when facing mounting flange end.						
Step	White	Blue	Red	Yellow	Brown	CW
1	On	On			+dcV	↓
2		On	On			
3			On	On		
4	On			On		

8 Wire configuration



Exciting sequence and direction of rotation when facing mounting flange end.

Step	Red	Green	Black	Yellow	Com	CW
1	On	On			+dcV	
2		On	On			
3			On	On		
4	On			On		

Technical specification

RS stock no.	440-420	440-436	440-442	440-458	440-464	440-470
Rated voltage (V)	5	12	5	12	3	2.5
Rated current (I)	0.5	0.16	1	0.6	2	4.5
Resistance (Ω)	10	75	5	20	1.5	0.56
Inductance (mH)	6	36	9	32	4.5	2.8
Detent torque (mHm)	5	4	30	30	40	100
Holding torque (mNm)	70	70	500	500	1200	2200
Step angle accuracy (%)	5	5	5	5	5	5
Step angle	1.8	1.8	1.8	1.8	1.8	1.8
Insulation class	B	B	B	B	B	B

RS stock no.	191-8299	191-8306	191-8328	191-8334	191-8340	191-8356	191-8362	191-8378	191-8384
Rated voltage (V)	12	15	5	12	12	12	5.4	3.4	6
Rated current (I)	0.4	0.4	1	0.4	0.48	0.6	1.4	2.85	1.8
Resistance (Ω)	30	45	5	40	25	20	3.8	1.2	3.5
Inductance (mH)	14	22	5.7	40	33	32	6.8	1.5	7.3
Detent torque (mHm)	3.5	3.5	14.8	14.8	29.6	29.6	56.5	77.6	77.6
Holding torque (mNm)	100	100	260	260	494	494	882	1200	1200
Step angle accuracy (%)	5	5	5	5	5	5	5	5	5
Step angle	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
Insulation class	B	B	B	B	B	B	B	B	B

Resonance

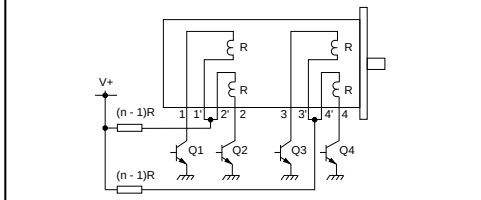
Certain operating frequencies cause resonance and the motor loses track of the drive input. Audible vibration may accompany resonance conditions. These frequencies should be avoided if possible. Driving the motor on the half step mode (see motor drive methods) greatly reduces the effect of resonance. Alternatively extra load inertia and external damping may be added to shift resonance regions away from the operating frequency.

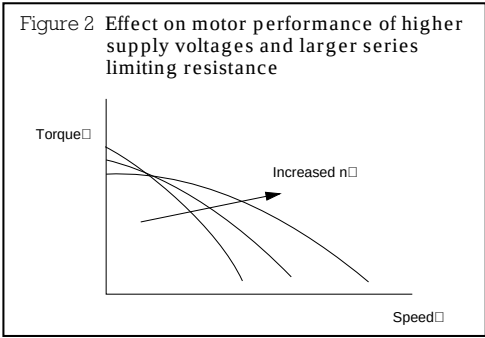
Motor drive methods

The normal way of driving a 4-phase stepper motor is shown in Figure 1.

This is commonly known as the 'Unipolar L/nr drive'. Here the current in each winding, when energised, flows in one direction only 'n', value is ≥ 1 (but not necessarily an integer) and nR is the sum of the external resistance plus the winding resistance (R). By selecting a higher value for n (ie. larger external resistance) and using a higher dc supply to maintain the rated voltage and current for each winding, improved torque speed characteristics can be obtained. Thus a 6V, 6 Ω motor (1A per phase) can be driven from a 6Vdc supply without any series resistor, in the L/R mode. Alternatively it can be driven from a 24Vdc supply using 18 Ω series resistance in the L/4R mode with much improved performance.

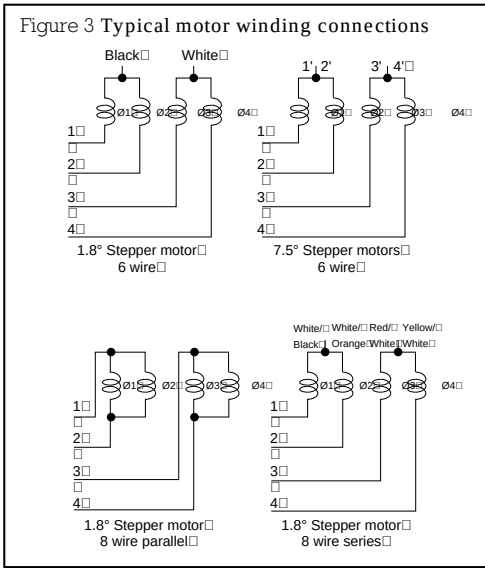
Figure 1 Unipolar drive





Connection to **RS** bipolar stepper motor board

When the windings of the RS stepper motors are assigned (Ø1-Ø4) as shown in Figure 3, they can be connected to the board according to Figure 1.



When using 8 lead motors with coils in parallel the motor current should be set no greater than:

$$I \text{ per phase} \times \sqrt{2}$$

When using 6 lead or 8 lead motors with coils in series the motor current should be set no greater than:

$$I \text{ per phase} \times \sqrt{\frac{1}{2}}$$

Motors with 4 leads have a bipolar rating and can be used according to manufacturer's specification.

To step a motor in a particular direction a specific switching sequence for the drive transistors Q1-Q4 needs to be followed. If this sequence is in Table 1 (known as the unipolar full step mode) it results in the rotor advancing through one complete step at a time.

Table 1 Full step mode

Step No.	Q1	Q2	Q3	Q4
Start position (arbitrary)	ON	OFF	OFF	ON
1	ON	OFF	ON	OFF
2	OFF	ON	ON	OFF
3	OFF	ON	OFF	ON
4	ON	OFF	OFF	ON
5	ON	OFF	ON	OFF

Above sequence repeating

Anti-clockwise

Clockwise

Table 2 Half step mode

Step No.	Q1	Q2	Q3	Q4
Start position	ON	OFF	ON	OFF
1	ON	OFF	OFF	OFF
2	ON	OFF	OFF	ON
3	OFF	OFF	OFF	ON
4	OFF	ON	OFF	ON
5	OFF	ON	OFF	OFF
6	OFF	ON	ON	OFF
7	OFF	OFF	ON	OFF
8	ON	OFF	ON	OFF
9				

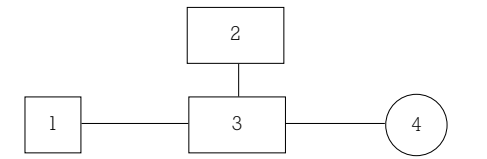
Above sequence repeating

Anti-clockwise

Clockwise

Typical stepper motor control system

The operation of a stepper motor requires the presence of the following elements:



1. **A control unit.** Usually a microprocessor based unit which gives step and direction signals to the drive card. RS stepper motor control board (RS stock no. 440-098) is ideally suited for this function.
2. **Power supply.** Giving the required voltage and current for the drive card using a linear power supply.
3. **Drive card.** This converts the signals from the control unit in to the required stepper motor sequence. RS stock nos. 332-098, 342-051 and 440-240 are designed for the function.
4. **Stepper motor.**

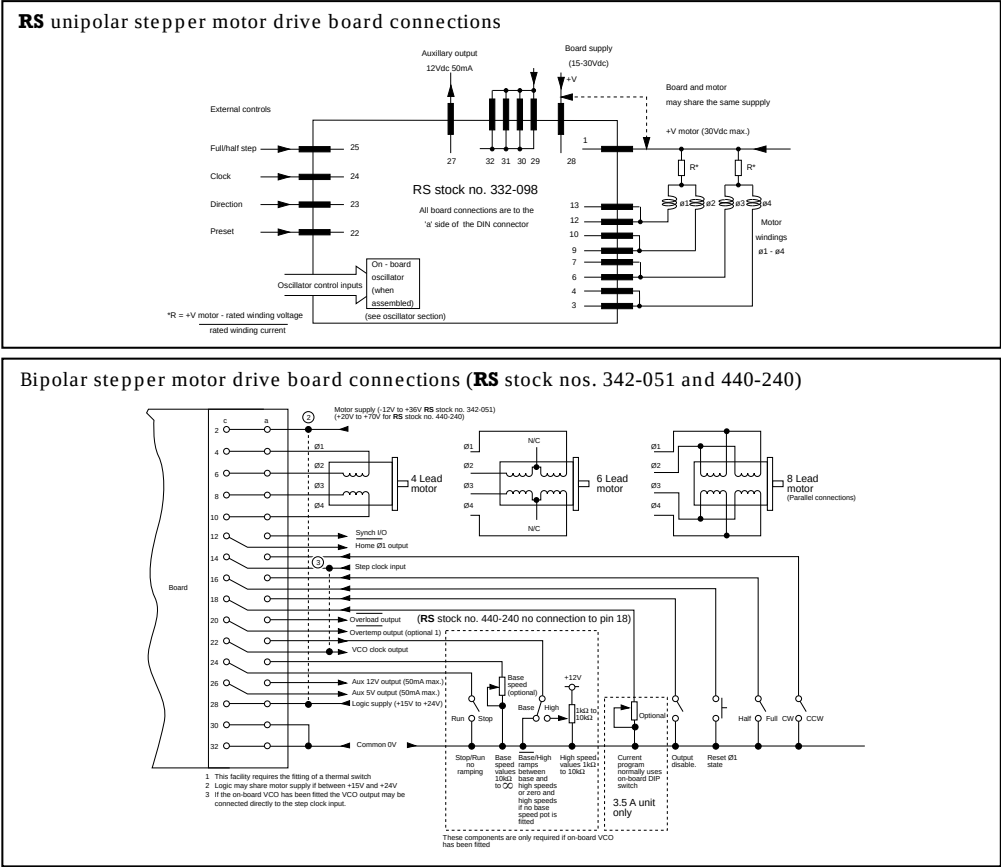
Stepper motor drive boards

For control of stepper motors RS has three types of stepper drive board which are suitable to drive stepper motors of various current ranges.

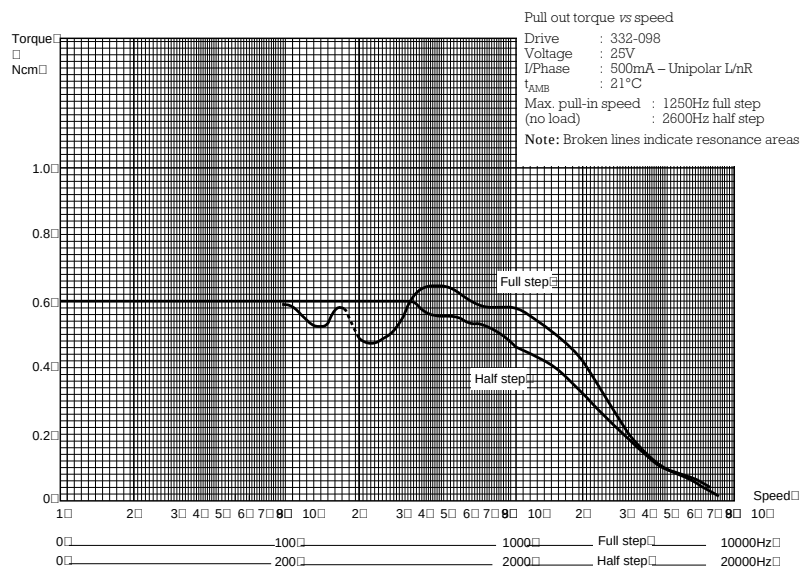
Drive board	RS stock nos.	Suitable stepper motors	Suggested wiring configuration
Unipolar 2A (RS stock no. 332-098) This drive is only suitable for applications where low speeds and low torques are required	440-420	Size 17	N/A
	440-436	Size 17	
	191-8299	Size 17	
	191-8306	Size 17	
	440-442	Size 23	
	440-458	Size 23	
	191-8328	Size 23	
	191-8334	Size 23	
	191-8340	Size 23	
	191-8356	Size 23	
	191-8362	Size 23	
	191-8384	Size 23	
Bipolar 3.5A (RS stock no. 342-051) Suitable for medium current, medium torque applications	440-442	Size 23	Series or parallel Parallel connection Series Series Series or parallel Series or parallel Series or parallel connection Series or parallel connection
	440-455	Size 23	
	191-8328	Size 23	
	191-8362	Size 23	
	191-8378	Size 23	
	191-8384	Size 23	
	440-464	Size 34	
Bipolar 6A (RS stock no. 440-240). Suitable for high current, high torque applications	191-8378	Size 23	Series or parallel Parallel Parallel connection Series or parallel connection
	191-8384	Size 23	
	440-464	Size 34	
	440-470	Size 34	

Note: Connecting a stepper motor in series will give a good low speed high torque performance.
Connecting a stepper motor in parallel will give a good high speed lower torque performance.

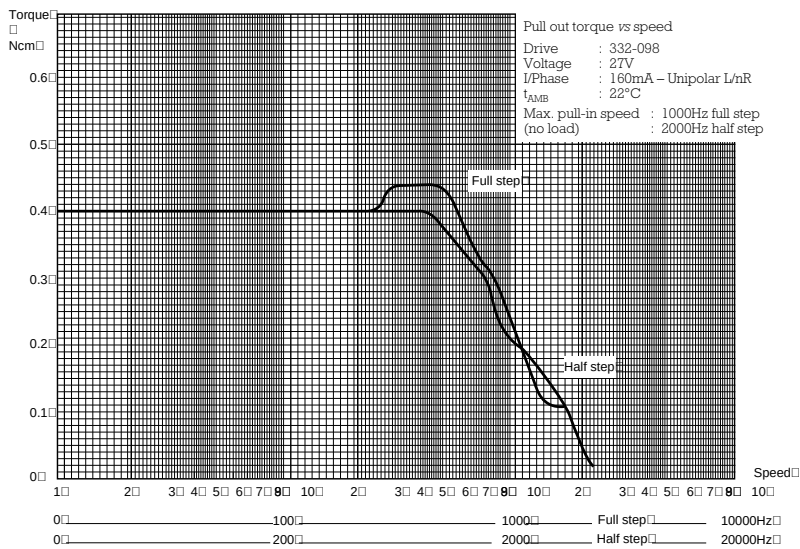
Drive board connections



Size 17 (5V 0.5A) (RS stock no. 440-420) using 2A unipolar drive board (RS stock no. 332-098)



Size 17 (12V 0.16A) (RS stock no. 440-436) using 2A unipolar drive board (RS stock no. 332-098)



The information provided in RS technical literature is believed to be accurate and reliable; however, RS Components assumes no responsibility for inaccuracies or omissions, or for the use of this information, and all use of such information shall be entirely at the user's own risk. No responsibility is assumed by RS Components for any infringements of patents or other rights of third parties which may result from its use. Specifications shown in RS Components technical literature are subject to change without notice.

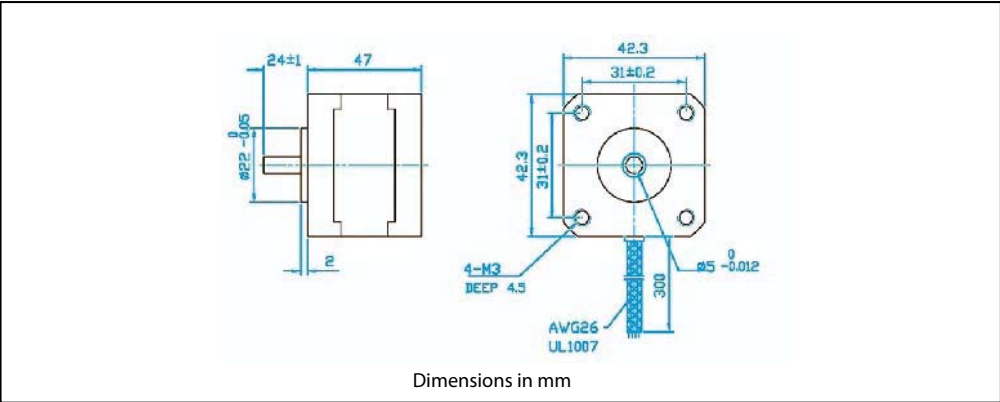
RS

42mm 1.8' HIGH TORQUE STEPPER

400 STEP

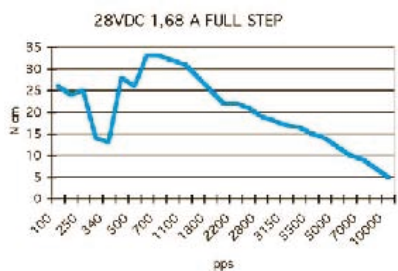
Data Sheet

RS stock number 535-0489



Specifications		Characteristics	
Model	535-0489	Resistance Accuracy	±10%
Step Angle	1.8°	Inductance Accuracy	±20%
Step Angle Accuracy (Full Step, No Load) (%)	±5%	Temperature Rise	80°C Max. (Rated Current 2 phase On)
Rated Voltage (V)	2.8	Ambient Temp	-20 °C to +50 °C
Current/ Phase (A)	1.68	Insulation Resistance	100mm Ωmin.,500Vdc
Resistance/Phase (Ω)	1.65	Dielectric Strength	500Vac for 1 min
Inductance/Phase (mH)	2.8	Shaft Radial Play	0.06mm Max. (450G-Load)
Detent Torque (mNm)	25	Shaft Axial Play	0.08mm Max. (450G-load)
Holding Torque (Ncm)	44		
Rotor Inertia (g·cm²)	68		
Weight (Kg)	0.35		
Number Of Leads (No.)	4		

Speed V Torque Characteristics



Size 16 hybrid stepper motor

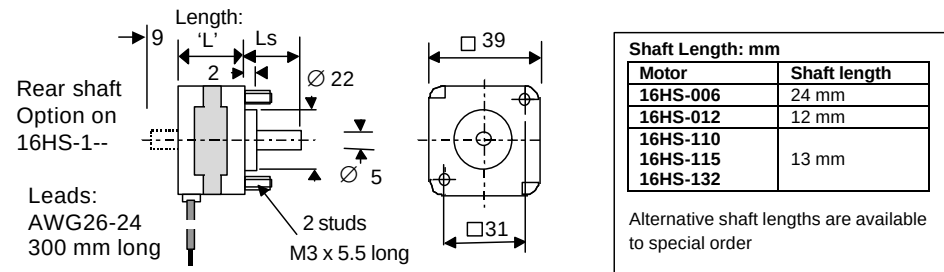
16 HS series

The 16 HS hybrid stepper motor provides 200 steps / revolution (400 steps /rev when operated in 1/2step mode) and is ideal for instrumentation drives requiring a combination of compact dimensions, high dynamic performance and accuracy. The motor is physically interchangeable with the larger 17HS model but it's reduced overall frame size and the reduced length of the 16HS-0 series makes it ideal for use in applications where space is at a premium.

The motor is available with a choice of windings to permit operation using either Uni-polar or Bi-polar drive circuits and offers an excellent combination of performance, reliability, and quality at economic prices ideally suited to OEM manufacturers of quality instrumentation products..



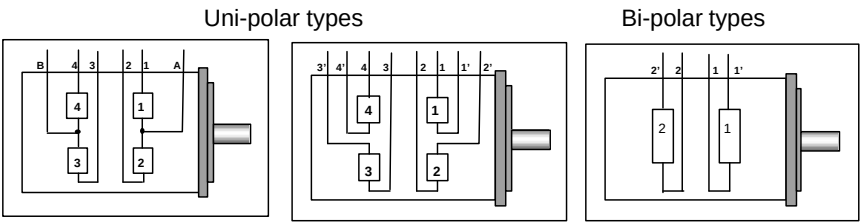
Dimensions: mm



Specification

motor type	length 'L' mm	holding torque Ncm	rotor inertia Kgcm ²	resistance per phase ohms	current per phase amps	inductance per phase mH	number of leads	mass Kg
Uni-polar types								
16HS-006	20.5	5	0.011	24	0.26	12.5	6	0.15
16HS-110	34	9	0.016	7.2	0.5	5.8	8	0.20
Bi-polar types								
16HS-012	20.5	8.7	0.011	6.6	0.6	8.5	4	0.15
16HS-115	34	11.5	0.016	4.4	0.75	5.6	4	0.20
16HS-132	34	12.0	0.016	1.2	1.6	1.4	4	0.20

Lead colours

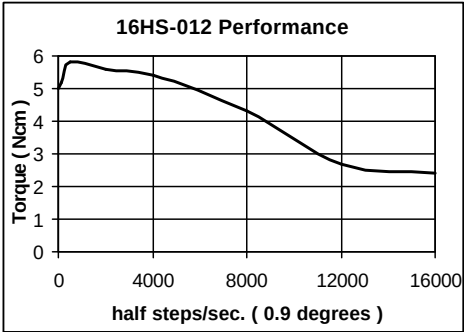


Lead colour identity:

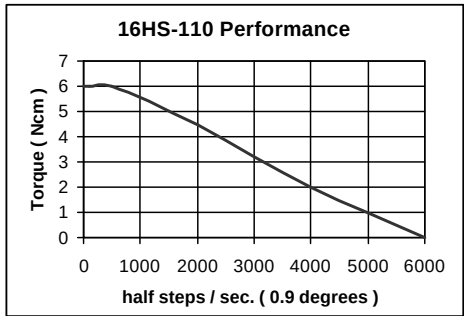
Uni-polar types		Bi-polar types	
6 lead versions		4 lead versions	
A	Black	1	Red
1	Red	1'	Yellow
2	Red / White	2	Black
B	White	2'	Orange
3	Green		
4	Green/White		
8 lead versions			
1'	Red/White		
1	Red		
2'	Yellow/white		
2	Yellow		
3'	Black/white		
3	Black		
4'	Orange/white		
4	Orange		

Typical 16HS Performance

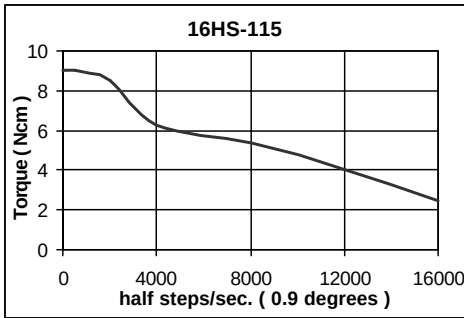
Bi-polar drive with 36 Vdc supply
0.6 Amps/phase



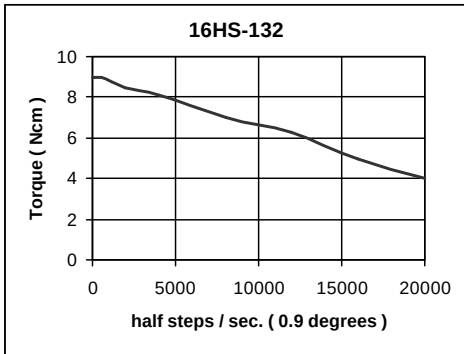
Uni-polar drive with 24Vdc supply
0.5 Amps /phase



Bi-polar drive with 36Vdc supply
0.75 Amps/phase



Bi-polar drive with 36Vdc supply
1.6 Amps/phase

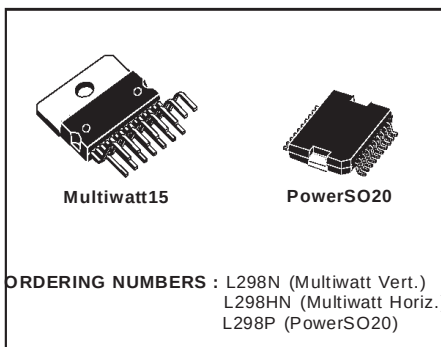


DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

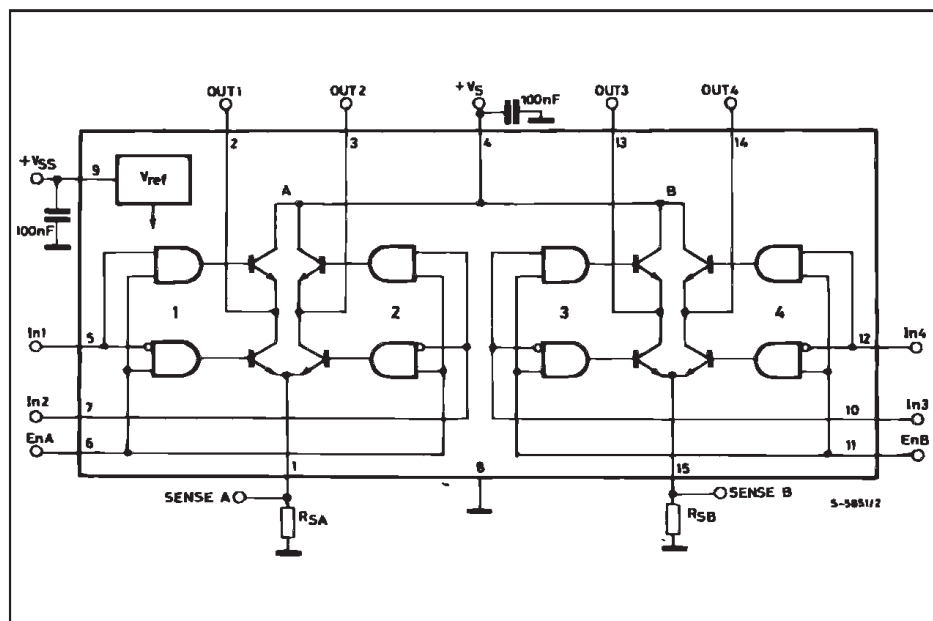
DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the con-



nection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

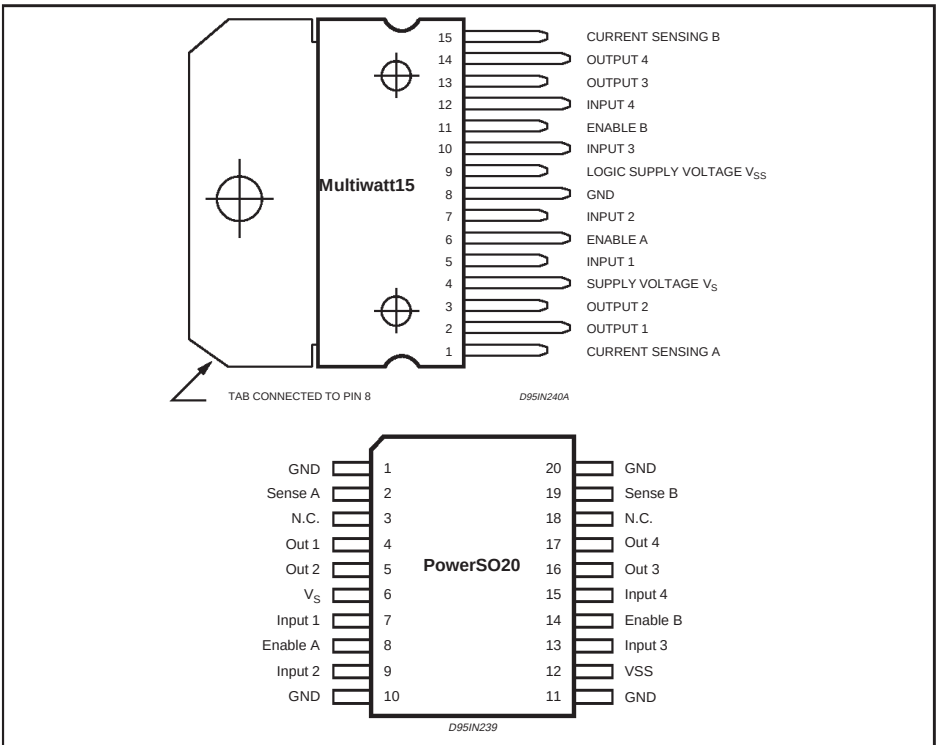
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{En}	Input and Enable Voltage	-0.3 to 7	V
I_O	Peak Output Current (each Channel)		
	– Non Repetitive ($t = 100\mu s$)	3	A
	– Repetitive (80% on –20% off; $t_{on} = 10ms$)	2.5	A
	– DC Operation	2	A
V_{sens}	Sensing Voltage	-1 to 2.3	V
P_{tot}	Total Power Dissipation ($T_{case} = 75^\circ C$)	25	W
T_{op}	Junction Operating Temperature	-25 to 130	$^\circ C$
T_{stg}, T_j	Storage and Junction Temperature	-40 to 150	$^\circ C$

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter	PowerSO20	Multiwatt15	Unit
$R_{th j-case}$	Thermal Resistance Junction-case	Max. –	3	$^\circ C/W$
$R_{th j-amb}$	Thermal Resistance Junction-ambient	Max. 13 (*)	35	$^\circ C/W$

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V _S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V _S	Supply Voltage (pin 4)	Operative Condition	V _{IH} +2.5		46	V
V _{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I _S	Quiescent Supply Current (pin 4)	V _{en} = H; I _L = 0 V _i = L V _i = H		13 50	22 70	mA mA
		V _{en} = L V _i = X			4	mA
I _{SS}	Quiescent Current from V _{SS} (pin 9)	V _{en} = H; I _L = 0 V _i = L V _i = H		24 7	36 12	mA mA
		V _{en} = L V _i = X			6	mA
V _{IL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V _{IH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		V _{SS}	V
I _{IL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V _i = L			–10	μA
I _{IH}	High Voltage Input Current (pins 5, 7, 10, 12)	V _i = H ≤ V _{SS} –0.6V		30	100	μA
V _{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V _{en} = H	Enable High Voltage (pins 6, 11)		2.3		V _{SS}	V
I _{en} = L	Low Voltage Enable Current (pins 6, 11)	V _{en} = L			–10	μA
I _{en} = H	High Voltage Enable Current (pins 6, 11)	V _{en} = H ≤ V _{SS} –0.6V		30	100	μA
V _{CEsat} (H)	Source Saturation Voltage	I _L = 1A I _L = 2A	0.95	1.35 2	1.7 2.7	V V
V _{CEsat} (L)	Sink Saturation Voltage	I _L = 1A (5) I _L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V _{CEsat}	Total Drop	I _L = 1A (5) I _L = 2A (5)	1.80		3.2 4.9	V V
V _{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$T_1 (V_i)$	Source Current Turn-off Delay	$0.5 V_i$ to $0.9 I_L$ (2); (4)		1.5		μs
$T_2 (V_i)$	Source Current Fall Time	$0.9 I_L$ to $0.1 I_L$ (2); (4)		0.2		μs
$T_3 (V_i)$	Source Current Turn-on Delay	$0.5 V_i$ to $0.1 I_L$ (2); (4)		2		μs
$T_4 (V_i)$	Source Current Rise Time	$0.1 I_L$ to $0.9 I_L$ (2); (4)		0.7		μs
$T_5 (V_i)$	Sink Current Turn-off Delay	$0.5 V_i$ to $0.9 I_L$ (3); (4)		0.7		μs
$T_6 (V_i)$	Sink Current Fall Time	$0.9 I_L$ to $0.1 I_L$ (3); (4)		0.25		μs
$T_7 (V_i)$	Sink Current Turn-on Delay	$0.5 V_i$ to $0.9 I_L$ (3); (4)		1.6		μs
$T_8 (V_i)$	Sink Current Rise Time	$0.1 I_L$ to $0.9 I_L$ (3); (4)		0.2		μs
$f_c (V_i)$	Commutation Frequency	$I_L = 2A$		25	40	KHz
$T_1 (V_{en})$	Source Current Turn-off Delay	$0.5 V_{en}$ to $0.9 I_L$ (2); (4)		3		μs
$T_2 (V_{en})$	Source Current Fall Time	$0.9 I_L$ to $0.1 I_L$ (2); (4)		1		μs
$T_3 (V_{en})$	Source Current Turn-on Delay	$0.5 V_{en}$ to $0.1 I_L$ (2); (4)		0.3		μs
$T_4 (V_{en})$	Source Current Rise Time	$0.1 I_L$ to $0.9 I_L$ (2); (4)		0.4		μs
$T_5 (V_{en})$	Sink Current Turn-off Delay	$0.5 V_{en}$ to $0.9 I_L$ (3); (4)		2.2		μs
$T_6 (V_{en})$	Sink Current Fall Time	$0.9 I_L$ to $0.1 I_L$ (3); (4)		0.35		μs
$T_7 (V_{en})$	Sink Current Turn-on Delay	$0.5 V_{en}$ to $0.9 I_L$ (3); (4)		0.25		μs
$T_8 (V_{en})$	Sink Current Rise Time	$0.1 I_L$ to $0.9 I_L$ (3); (4)		0.1		μs

1) Sensing voltage can be $-1 V$ for $t \leq 50 \mu s$; in steady state $V_{sens} \min \geq -0.5 V$.

2) See fig. 2.

3) See fig. 4.

4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

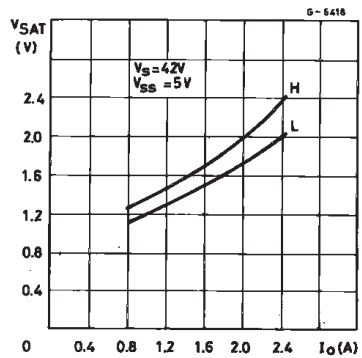
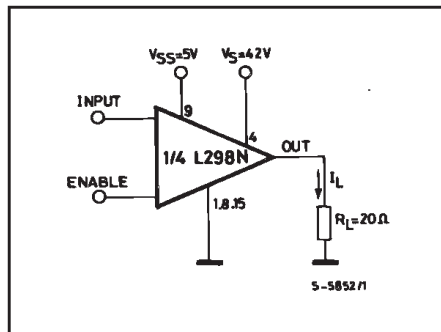
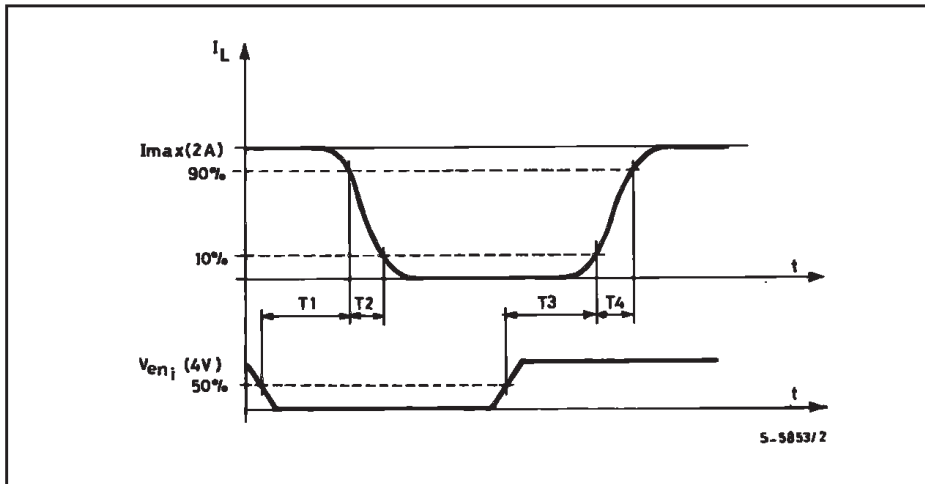
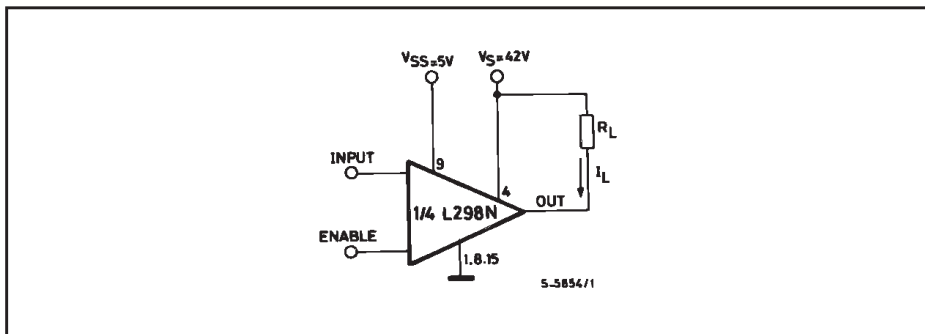


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.**Figure 4** : Switching Times Test Circuits.

Note : For INPUT Switching, set EN = H
 For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

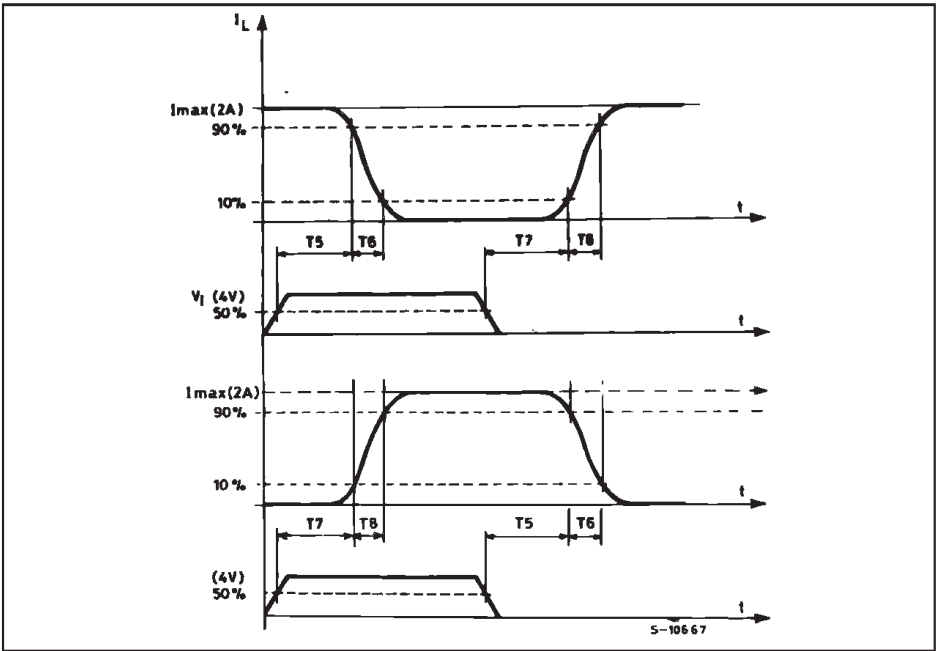


Figure 6 : Bidirectional DC Motor Control.

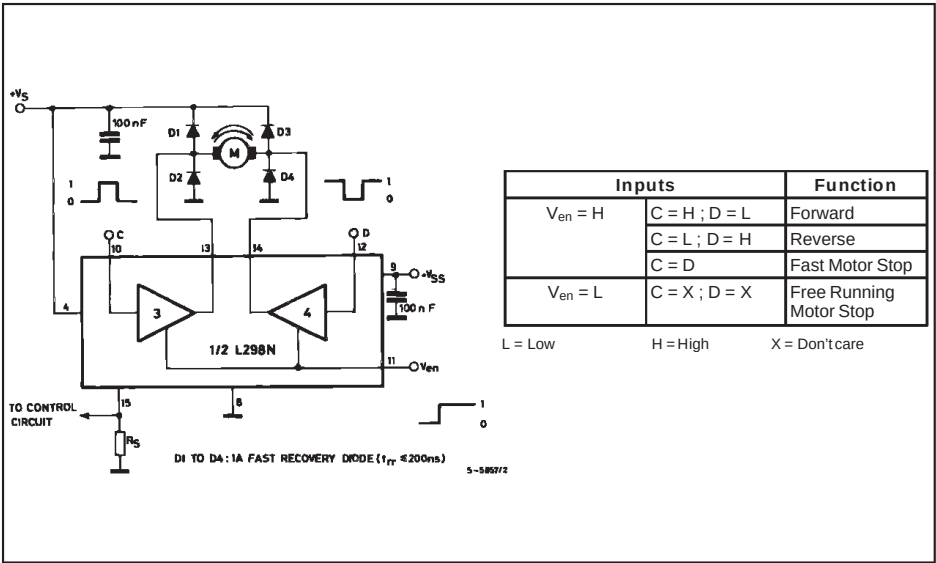
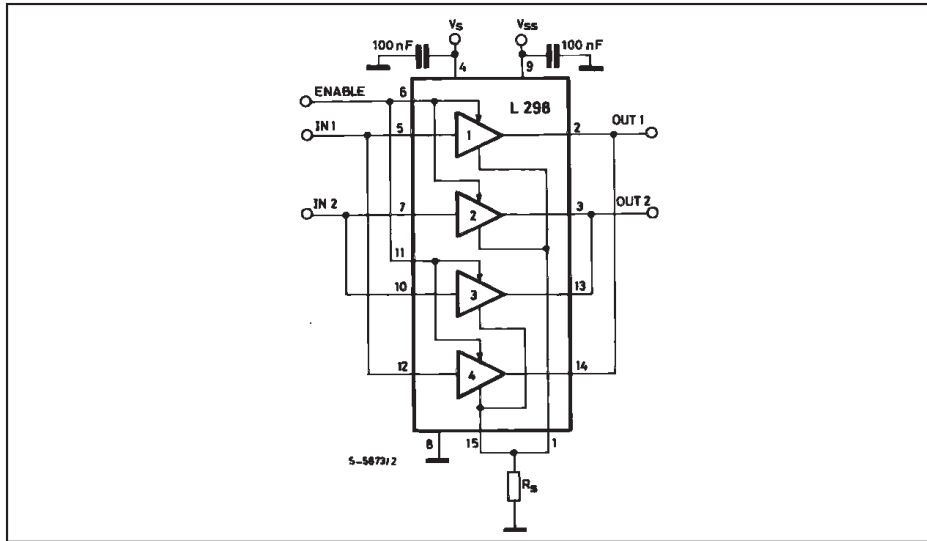


Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differential mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output: an external resistor (R_{SA} ; R_{SB}) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are $IN1$; $IN2$; EN_A and $IN3$; $IN4$; EN_B . The IN inputs set the bridge state when the EN input is high; a low state of the EN input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both V_S and V_{SS} , to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of V_S that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes $D1$ to $D4$ is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped; Schottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.

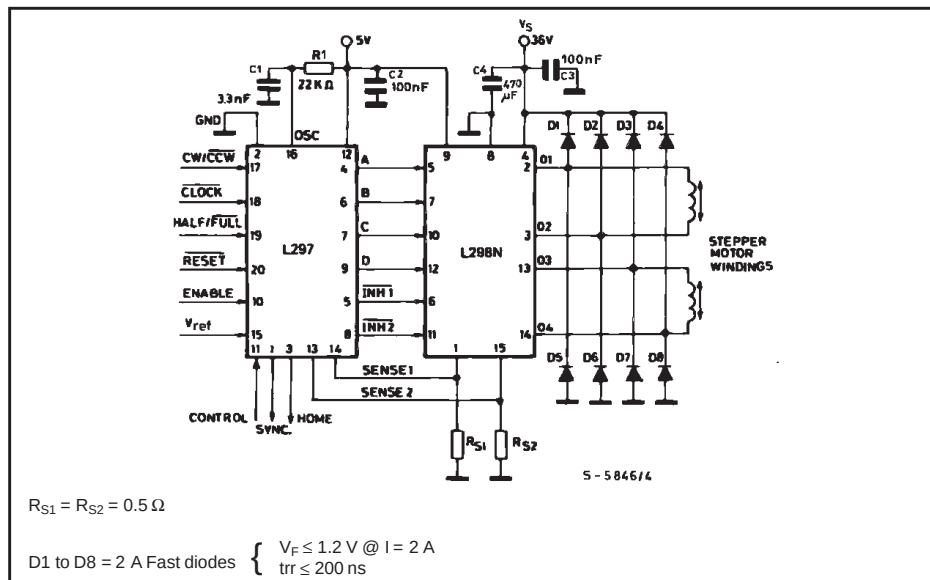


Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

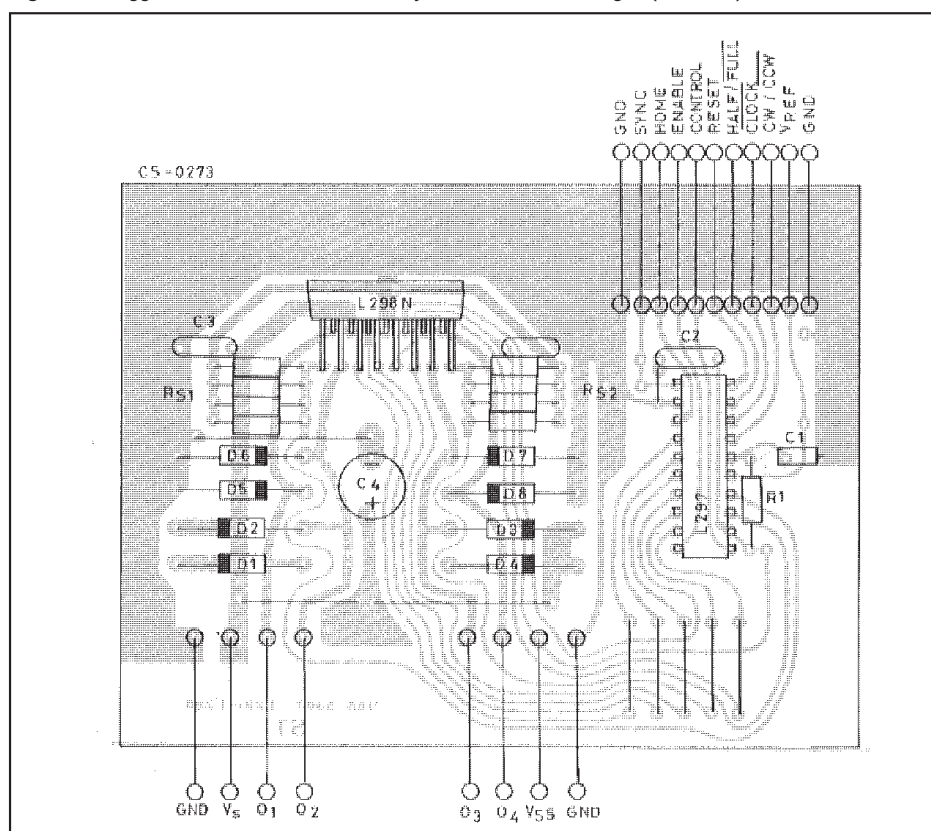
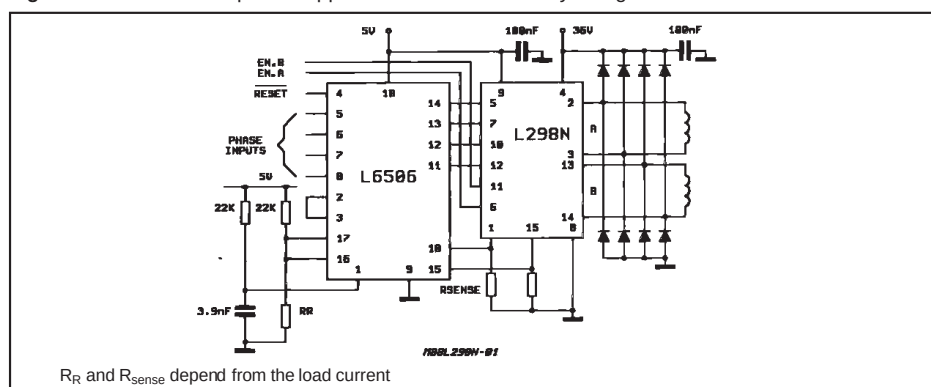


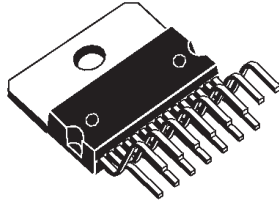
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



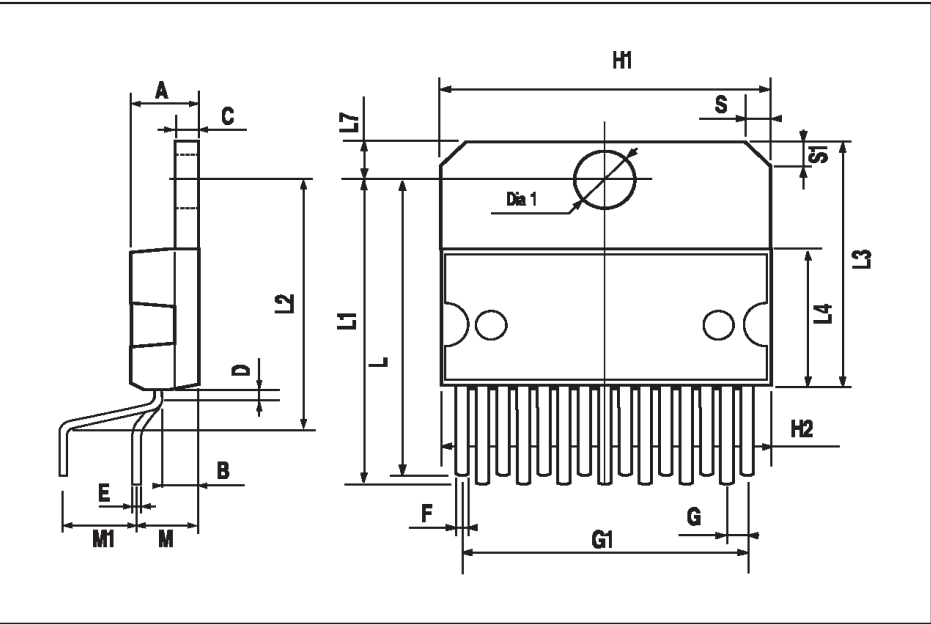
L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

**OUTLINE AND
MECHANICAL DATA**

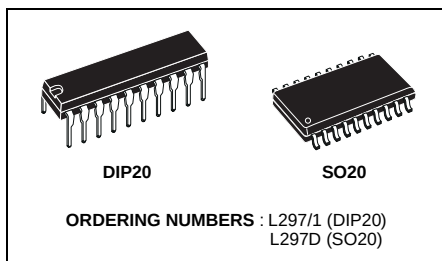


Multiwatt15 V



STEPPER MOTOR CONTROLLERS

- NORMAL/WAVE DRIVE
- HALF/FULL STEP MODES
- CLOCKWISE/ANTICLOCKWISE DIRECTION
- SWITCHMODE LOAD CURRENT REGULATION
- PROGRAMMABLE LOAD CURRENT
- FEW EXTERNAL COMPONENTS
- RESET INPUT & HOME OUTPUT
- ENABLE INPUT



DESCRIPTION

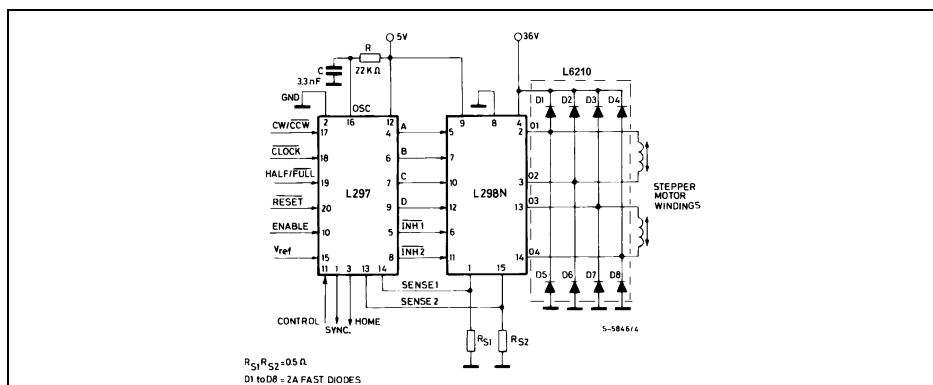
The L297 Stepper Motor Controller IC generates four phase drive signals for two phase bipolar and four phase unipolar step motors in microcomputer-controlled applications. The motor can be driven in half step, normal and wave drive modes and on-chip PWM chopper circuits permit switch-mode control of the current in the windings. A feature of

this device is that it requires only clock, direction and mode input signals. Since the phase are generated internally the burden on the microprocessor, and the programmer, is greatly reduced. Mounted in DIP20 and SO20 packages, the L297 can be used with monolithic bridge drives such as the L298N or L293E, or with discrete transistors and darlingtonts.

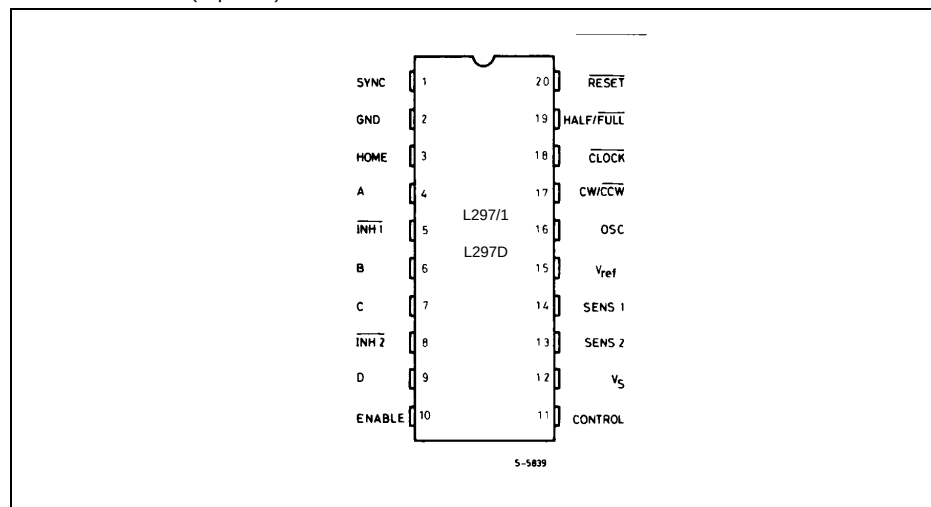
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply voltage	10	V
V_i	Input signals	7	V
P_{tot}	Total power dissipation ($T_{amb} = 70^\circ\text{C}$)	1	W
T_{stg}, T_j	Storage and junction temperature	-40 to + 150	$^\circ\text{C}$

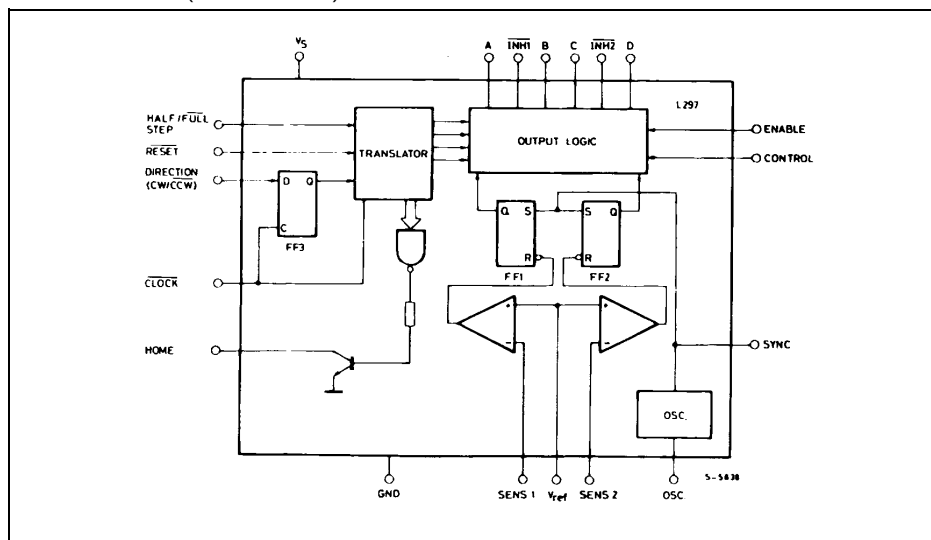
TWO PHASE BIPOLAR STEPPER MOTOR CONTROL CIRCUIT



PIN CONNECTION (Top view)



BLOCK DIAGRAM (L297/1 - L297D)



PIN FUNCTIONS - L297/1 - L297D

N°	NAME	FUNCTION
1	SYNC	Output of the on-chip chopper oscillator. The SYNC connections of all L297s to be synchronized are connected together and the oscillator components are omitted on all but one. If an external clock source is used it is injected at this terminal.
2	GND	Ground connection.
3	HOME	Open collector output that indicates when the L297 is in its initial state (ABCD = 0101). The transistor is open when this signal is active.
4	A	Motor phase A drive signal for power stage.
5	$\overline{\text{INH1}}$	Active low inhibit control for driver stage of A and B phases. When a bipolar bridge is used this signal can be used to ensure fast decay of load current when a winding is de-energized. Also used by chopper to regulate load current if CONTROL input is low.
6	B	Motor phase B drive signal for power stage.
7	C	Motor phase C drive signal for power stage.
8	$\overline{\text{INH2}}$	Active low inhibit control for drive stages of C and D phases. Same functions as INH1.
9	D	Motor phase D drive signal for power stage.
10	ENABLE	Chip enable input. When low (inactive) INH1, INH2, A, B, C and D are brought low.
11	CONTROL	Control input that defines action of chopper. When low chopper acts on INH1 and INH2; when high chopper acts on phase lines ABCD.
12	V _s	5V supply input.
13	SENS ₂	Input for load current sense voltage from power stages of phases C and D.
14	SENS ₁	Input for load current sense voltage from power stages of phases A and B.
15	V _{ref}	Reference voltage for chopper circuit. A voltage applied to this pin determines the peak load current.
16	OSC	An RC network (R to V _{CC} , C to ground) connected to this terminal determines the chopper rate. This terminal is connected to ground on all but one device in synchronized multi - L297 configurations. $f \approx 1/0.69 RC$
17	$\overline{\text{CW/CCW}}$	Clockwise/counterclockwise direction control input. Physical direction of motor rotation also depends on connection of windings. Synchronized internally therefore direction can be changed at any time.
18	$\overline{\text{CLOCK}}$	Step clock. An active low pulse on this input advances the motor one increment. The step occurs on the rising edge of this signal.

PIN FUNCTIONS - L297/1 - L297D (continued)

N°	NAME	FUNCTION
19	HALF/FULL	Half/full step select input. When high selects half step operation, when low selects full step operation. One-phase-on full step mode is obtained by selecting FULL when the L297's translator is at an even-numbered state. Two-phase-on full step mode is set by selecting FULL when the translator is at an odd numbered position. (The home position is designate state 1).
20	RESET	Reset input. An active low pulse on this input restores the translator to the home position (state 1, ABCD = 0101).

THERMAL DATA

Symbol	Parameter	DIP20	SO20	Unit
R _{th-j-amb}	Thermal resistance junction-ambient	80	100	°C/W

CIRCUIT OPERATION

The L297 is intended for use with a dual bridge driver, quad darlington array or discrete power devices in step motor driving applications. It receives step clock, direction and mode signals from the systems controller (usually a microcomputer chip) and generates control signals for the power stage.

The principal functions are a translator, which generates the motor phase sequences, and a dual PWM chopper circuit which regulates the current in the motor windings. The translator generates three different sequences, selected by the HALF/FULL input. These are normal (two phases energised), wave drive (one phase energised) and half-step (alternately one phase energised/two phases energised). Two inhibit signals are also generated by the L297 in half step and wave drive modes. These signals, which connect directly to the L298's enable inputs, are intended to speed current decay when a winding is de-energised. When the L297 is used to drive a unipolar motor the chopper acts on these lines.

An input called CONTROL determines whether the chopper will act on the phase lines ABCD or the inhibit lines INH1 and INH2. When the phase lines

are chopped the non-active phase line of each pair (AB or CD) is activated (rather than interrupting the line then active). In L297 + L298 configurations this technique reduces dissipation in the load current sense resistors.

A common on-chip oscillator drives the dual chopper. It supplies pulses at the chopper rate which set the two flip-flops FF1 and FF2. When the current in a winding reaches the programmed peak value the voltage across the sense resistor (connected to one of the sense inputs SENS₁ or SENS₂) equals V_{ref} and the corresponding comparator resets its flip flop, interrupting the drive current until the next oscillator pulse arrives. The peak current for both windings is programmed by a voltage divider on the V_{ref} input.

Ground noise problems in multiple configurations can be avoided by synchronising the chopper oscillators. This is done by connecting all the SYNC pins together, mounting the oscillator RC network on one device only and grounding the OSC pin on all other devices.

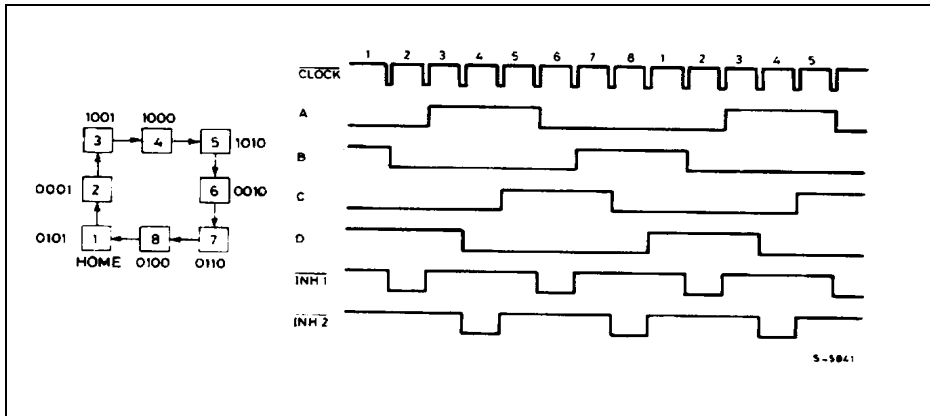
MOTOR DRIVING PHASE SEQUENCES

The L297's translator generates phase sequences for normal drive, wave drive and half step modes. The state sequences and output waveforms for these three modes are shown below. In all cases the translator advances on the low to high transition of $\overline{\text{CLOCK}}$.

Clockwise rotation is indicated; for anticlockwise rotation the sequences are simply reversed. $\overline{\text{RESET}}$ restores the translator to state 1, where ABCD = 0101.

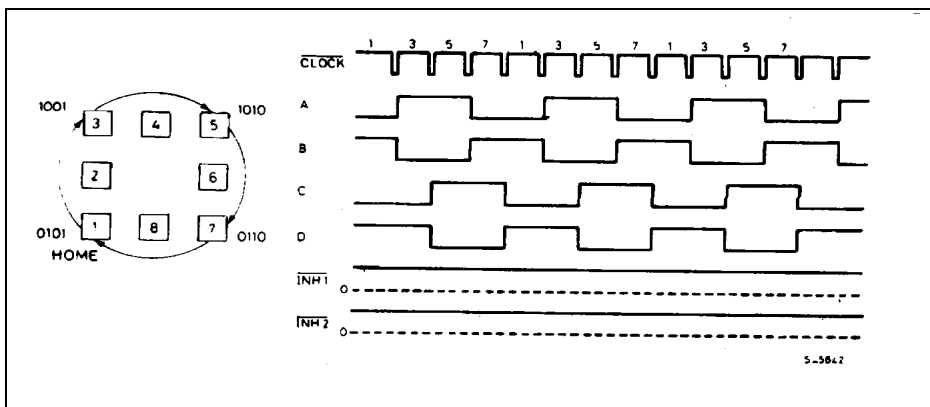
HALF STEP MODE

Half step mode is selected by a high level on the $\overline{\text{HALF/FULL}}$ input.



NORMAL DRIVE MODE

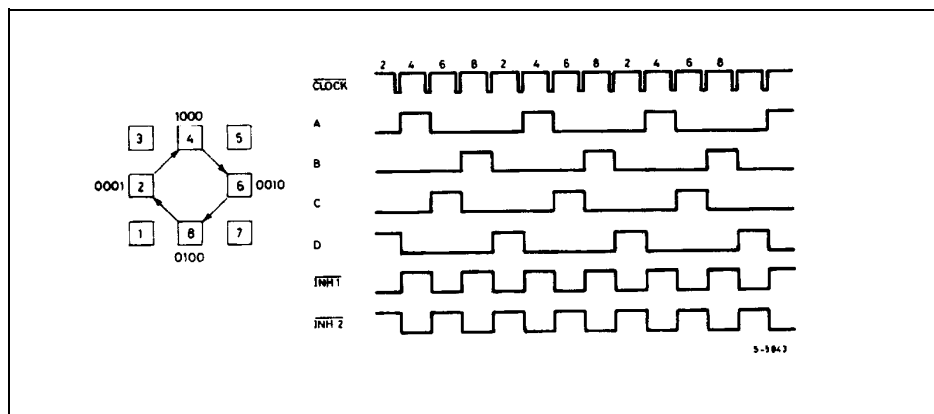
Normal drive mode (also called "two-phase-on" drive) is selected by a low level on the $\overline{\text{HALF/FULL}}$ input when the translator is at an odd numbered state (1, 3, 5 or 7). In this mode the $\overline{\text{INH1}}$ and $\overline{\text{INH2}}$ outputs remain high throughout.



MOTOR DRIVING PHASE SEQUENCES (continued)

WAVE DRIVE MODE

Wave drive mode (also called "one-phase-on" drive) is selected by a low level on the HALF/FULL input when the translator is at an even numbered state (2, 4, 6 or 8).

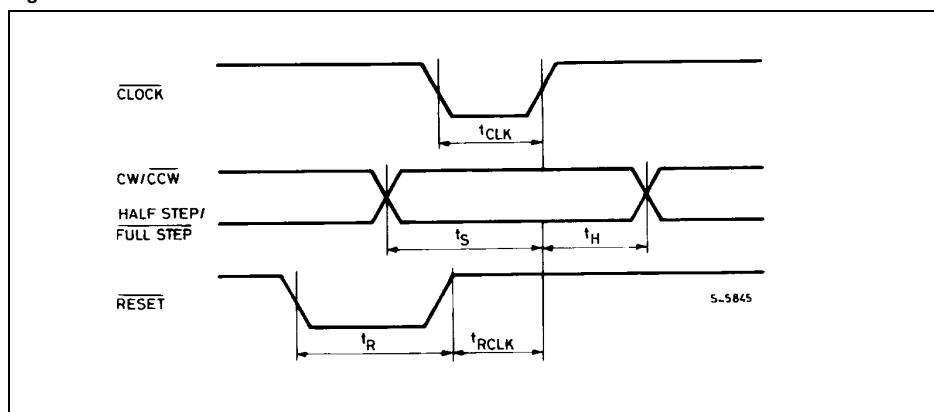
ELECTRICAL CHARACTERISTICS (Refer to the block diagram $T_{amb} = 25^{\circ}\text{C}$, $V_s = 5\text{V}$ unless otherwise specified)

Symbol	Parameter	Test conditions	Min.	Typ	Max.	Unit
V_s	Supply voltage (pin 12)		4.75		7	V
I_s	Quiescent supply current (pin 12)	Outputs floating		50	80	mA
V_i	Input voltage (pin 11, 17, 18, 19, 20)	Low			0.6	V
		High	2		V_s	V
I_i	Input current (pin 11, 17, 18, 19, 20)	$V_i = L$		100		μA
		$V_i = H$			10	μA
V_{en}	Enable input voltage (pin 10)	Low			1.3	V
		High	2		V_s	V
I_{en}	Enable input current (pin 10)	$V_{en} = L$			100	μA
		$V_{en} = H$			10	μA
V_o	Phase output voltage (pins 4, 6, 7, 9)	$I_o = 10\text{mA}$ V_{OL}			0.4	V
		$I_o = 5\text{mA}$ V_{OH}	3.9			V
V_{inh}	Inhibit output voltage (pins 5, 8)	$I_o = 10\text{mA}$ $V_{inh L}$			0.4	V
		$I_o = 5\text{mA}$ $V_{inh H}$	3.9			V
V_{SYNC}	Sync Output Voltage	$I_o = 5\text{mA}$ $V_{SYNC H}$	3.3			V
		$I_o = 5\text{mA}$ $V_{SYNC V}$			0.8	

ELECTRICAL CHARACTERISTICS (continued)

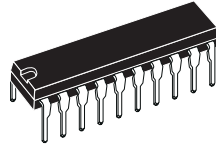
Symbol	Parameter	Test conditions	Min.	Typ	Max.	Unit
I_{leak}	Leakage current (pin 3)	$V_{CE} = 7\text{ V}$			1	μA
V_{sat}	Saturation voltage (pin 3)	$I = 5\text{ mA}$			0.4	V
V_{off}	Comparators offset voltage (pins 13, 14, 15)	$V_{ref} = 1\text{ V}$			5	mV
I_o	Comparator bias current (pins 13, 14, 15)		-100		10	μA
V_{ref}	Input reference voltage (pin 15)		0		3	V
t_{CLK}	Clock time		0.5			μs
t_s	Set up time		1			μs
t_H	Hold time		4			μs
t_R	Reset time		1			μs
t_{RCLK}	Reset to clock delay		1			μs

Figure 1.

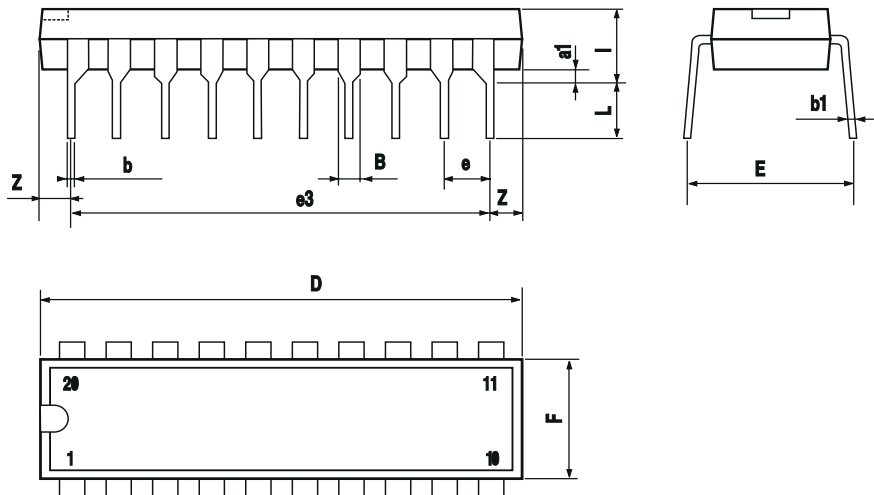


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.254			0.010		
B	1.39		1.65	0.055		0.065
b		0.45			0.018	
b1		0.25			0.010	
D			25.4			1.000
E		8.5			0.335	
e		2.54			0.100	
e3		22.86			0.900	
F			7.1			0.280
l			3.93			0.155
L		3.3			0.130	
Z			1.34			0.053

OUTLINE AND MECHANICAL DATA

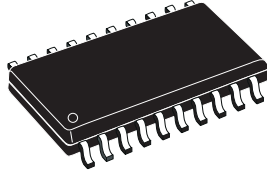


DIP20

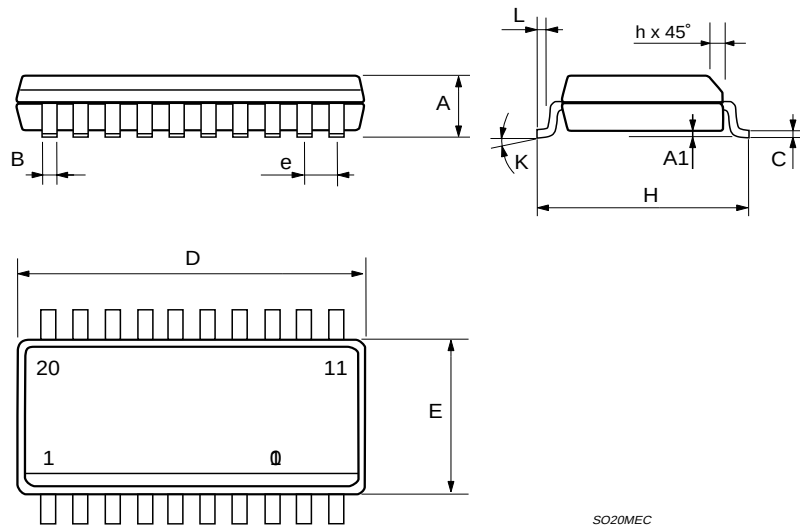


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	2.35		2.65	0.093		0.104
A1	0.1		0.3	0.004		0.012
B	0.33		0.51	0.013		0.020
C	0.23		0.32	0.009		0.013
D	12.6		13	0.496		0.512
E	7.4		7.6	0.291		0.299
e		1.27			0.050	
H	10		10.65	0.394		0.419
h	0.25		0.75	0.010		0.030
L	0.4		1.27	0.016		0.050
K	0° (min.)8° (max.)					

OUTLINE AND
MECHANICAL DATA



SO20



SO20MEC

